TissUUmaps

Release 3.0

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

Sep 25, 2023

CONTENTS:

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

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

Work in progress!

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
   conda activate tissuumaps_env
   ```

2. Install dependencies using `conda`:

   ```
   conda install -c conda-forge libvips pyvips openslide-python
   ```

3. Install the TissUUmaps library using `pip`:

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

   Note: If the installation fails with PyQt6, you can remove `[full]` from the previous command and run step 5 to start TissUUmaps server.

4. Start the TissUUmaps user interface:

   ```
   tissuumaps
   ```

5. 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 3.1.1.6

- Add a min-max filter to image layers for better contrast (@glbarlow)
- Fix marker filtering
- Update license from BSD-3 to MIT
- Add collapsible sections in plugins
- Minor fixes

1.4.2 3.1.1.5

- Fix multiple images dropped on empty project
- Change threading to allow higher priority to GUI than to server side
- Add .tmap extension to all saved projects
- Fix pie charts display update and colors
- Add error message if update view fails

1.4.3 3.1.1.4

- Fix bug of modal not hiding properly
- Fix capture with scale of 1
- Fix loading of AnnData files without X matrix
- Add DEFAULT_PROJECT parameter for docker and server

1.4.4 3.1.1.3

- Minor fixes

1.4.5 3.1.1.2

- Make load of HDF5 data parallel
- Add Colormaps and Channel splitting filters

1.4.6 3.1.1

- Move to OpenSeadragon 4.0.0
- Move docker to python-alpine for security reasons
- Add sorting options of markers (applied automatically on AnnData observations)
- Add zOrder parameter for draw order between datasets
- Make Update View button always visible
- Add colorbar canvas to png captures
- Add flip and rotation of markers for each collection item
- Add transformation inputs in Image Layers menu
- Add background color input
- Add gaussian marker shape
- Fix flask deprecated function
- Minor fixes

1.4.7 3.1.0.8

- Fix chunk size for h5 files
- Fix iPhone not loading markers with > 192 collection items

1.4.8 3.1.0.7

- Critical fix for Mac OS (see row_major in WebGL2)

1.4.9 3.1.0.6

- Minor fixes

1.4.10 3.1.0.5

- Add flip and rotation of markers for each collection item
- Add transformation inputs in Image Layers menu
- Fix colormap with NaN values

1.4.11 3.1.0.4

- Fix issue with slow rendering of web-gl using instancing
- Hide colorbar on png captures when needed
- Other minor fixes

1.4.12 3.1.0.3

- Fix h5 autocomplete on iframes
- Fix wrong marker index for color column when more than 100k markers
- Add colorbar canvas to png captures
- Small fixes

1.4.13 3.1.0.2

- Fix crash on layer from a parent layer.
- Change dropdown selection from Chosen to Select2 for faster loading.
- Update docker to Alpine for security reasons.
- Small fixes

1.4.14 3.1.0.1

- Move docker to python-alpine for security reasons
- Add sorting options of markers (applied automatically on AnnData observations)
- Make Update View button always visible
- Minor fixes

1.4.15 3.1

- Adding HDF5 support on the client side
- Adding AnnData support on the server side / GUI
- Adding Network diagram visualization
- Tabs now saved automatically even without buttons
- Adding Plugin helpers in javascript
- Many fixes on the interface
- Move to PyQt6

1.4.16 3.0.10.4

- Fix path issue on json loading from server

1.4.17 3.0.10.3

- Reset all input dropdowns when new data selected

1.4.18 3.0.10.2

- Add scale factor for coordinates of markers

1.4.19 3.0.10.1

- Add optional offset (x, y) and scale properties to tmap.layers

1.4.20 3.0.10

- Add collection mode (to display images next to each other with markers correctly placed)
- IFrame mode (to hide navbar and make menu smaller when TissUUmaps is run inside an iFrame)

1.4.21 3.0.9.6

- Add debug menu when running in debug mode, with debug access in javascript
- Fix linux bugs with Qt displaying all blank
- Fix empty columns in marker csv file

1.4.22 3.0.9.5

- Add / fix key shortcuts (https://tissuumaps.github.io/TissUUmaps-docs/docs/starting/shortcuts.html)
- Change default GUI port to avoid collisions with server
- Add plugin support to docker server

1.4.23 3.0.9.3

- Go back to webGL 1 for compatibility issue with Safari 14
- Fix missing .tissuumaps folder for recent files

1.4.24 3.0.9.1

- Enable larger markers at high resolutiion (up to 1024x1024px)
- Fix pinch to zoom center
- Add code of conduct
- Clean code and use ci (pre-commit)

1.4.25 3.0.9

- Move to webgl2
- Add Open Recent sub menu in File menu
- Fix path for linux and mac in server mode
- Minor fixes

1.4.26 3.0.8.9

- Make it possible to update to newer version of plugins
- Fully support –debug option in command line
- Add tooltip title for piecharts
- Add documentation https://tissuumaps.github.io/TissUUmaps-docs/
- Fix marker picking when pixel ratio != 1
- Other minor fixes and cleaning

1.4.27 3.0.8.5

- Minor fixes.

1.4.28 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.

1.4.29 3.0.8.3

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

1.4.30 3.0.8.2

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

1.4.31 3.0.8.1

- Fix multiple dataset alignment when no background image

1.4.32 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

1.4.33 3.0.7

- Add menu to load plugins through an update-site

1.4.34 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

1.4.35 3.0.5

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

1.4.36 3.0.4

- Add filtering of markers

1.4.37 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:

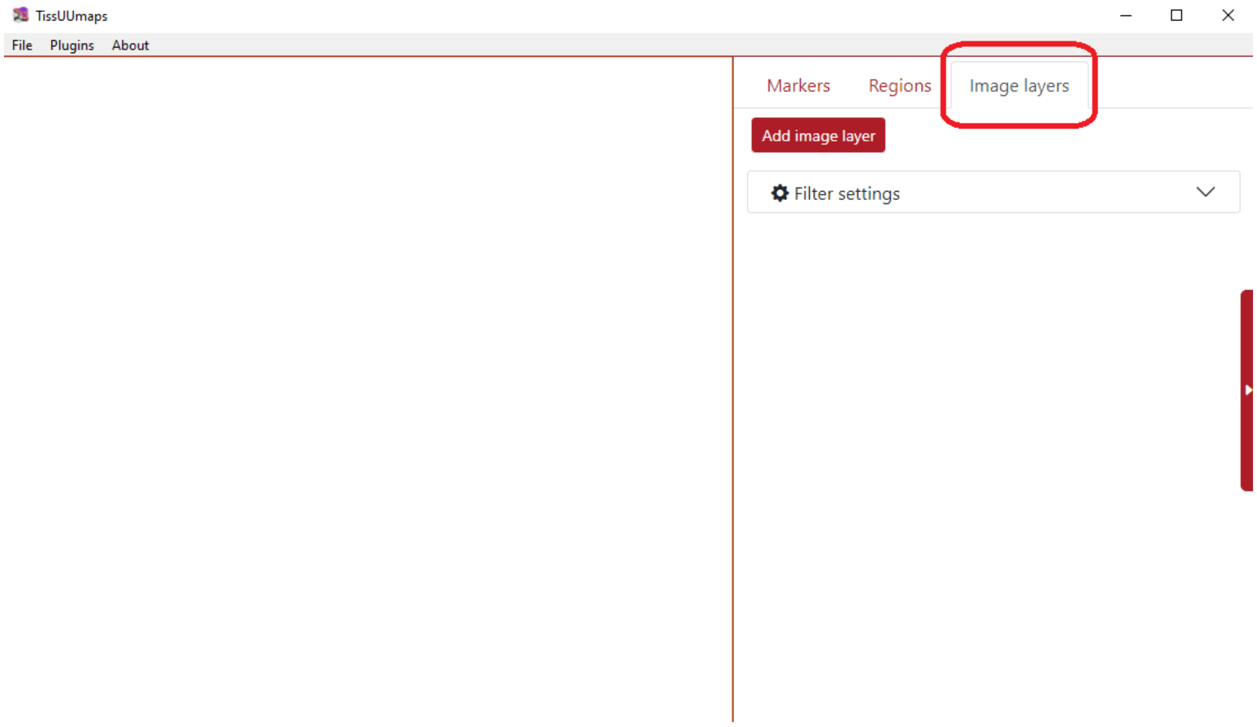
- Aperio (.svs, .tif)
- Hamamatsu (.ndpi, .vms, .vmu)
- Leica (.scn)
- MIRAX (.mrxs)
- Philips (.tiff)
- Sakura (.svslide)
- Trestle (.tif)
- Ventana (.bif, .tif)
- Generic tiled TIFF (.tif)

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

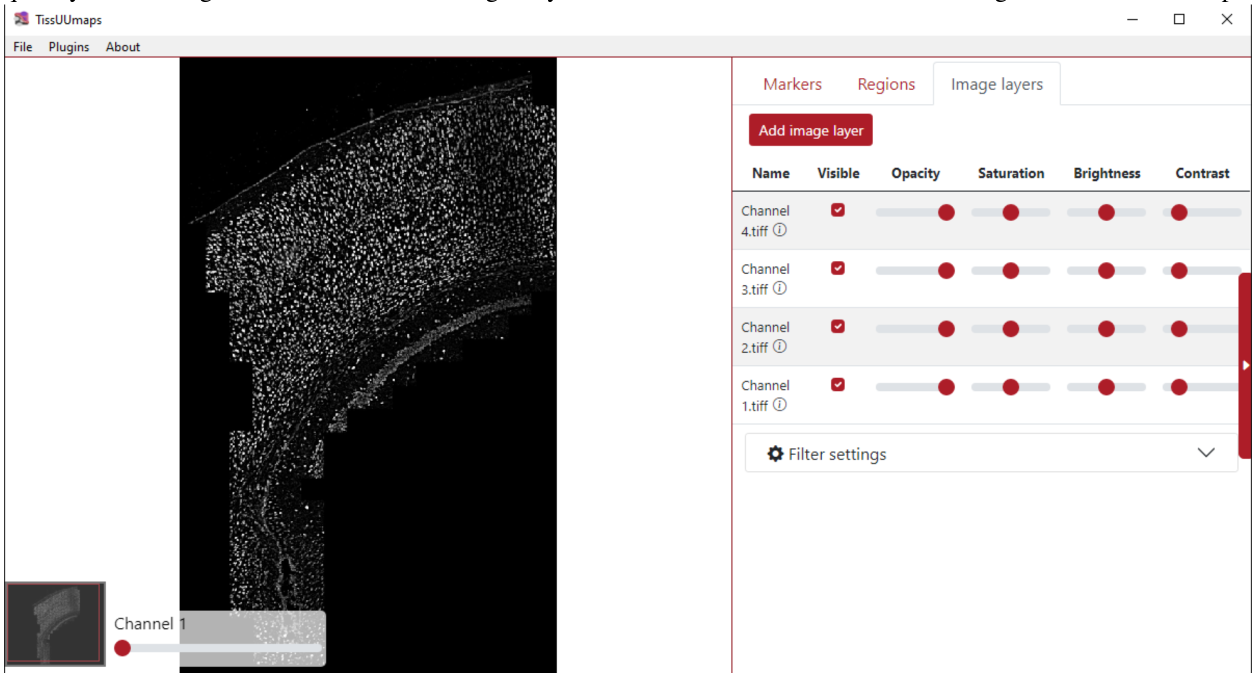
If your image fails to open, try converting it to `tif` format using an external tool.

2.1.2 Load images

You can load the images when you select the Image layer tab as you can see in the figure below:

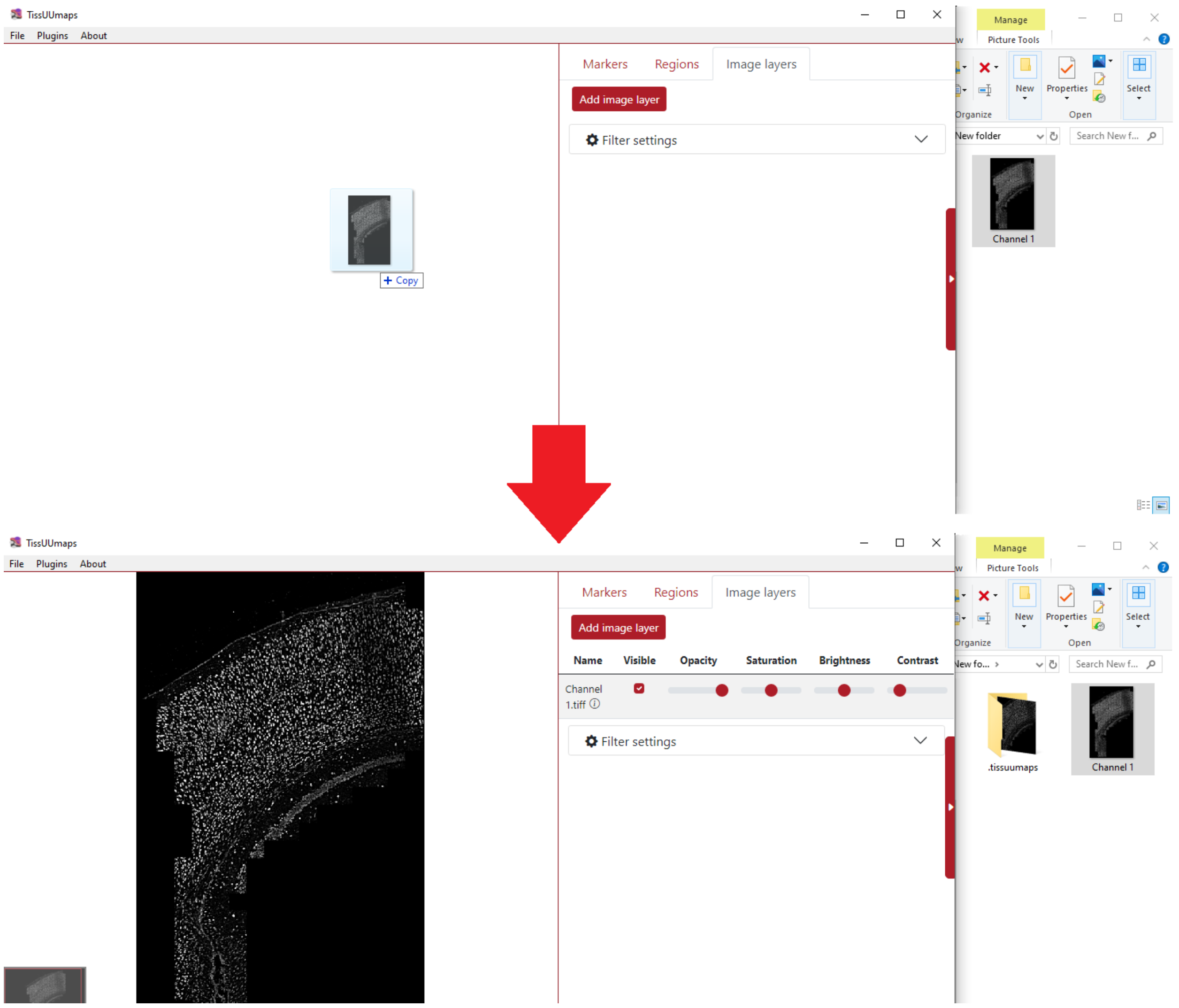


Then click the button Add image layer and select the desired image from your computer. Subsequently, the image is listed in the Image layer tab. You can load several images into TissUUmaps.



Name	Visible	Opacity	Saturation	Brightness	Contrast
Channel 4.tiff	✓	●	●	●	●
Channel 3.tiff	✓	●	●	●	●
Channel 2.tiff	✓	●	●	●	●
Channel 1.tiff	✓	●	●	●	●

You can also drag and drop the image from file explorer into TissUUmaps.

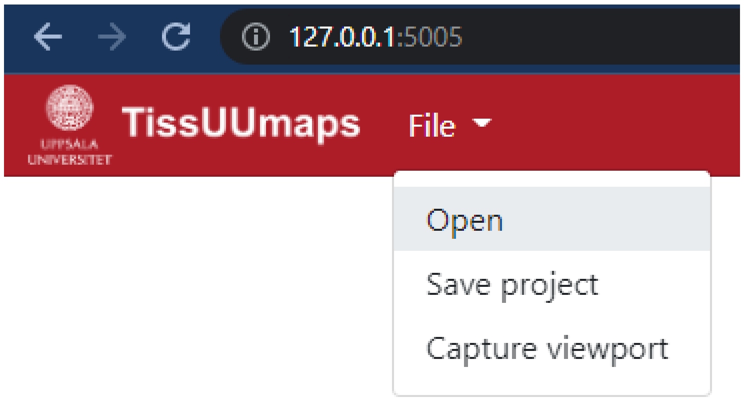


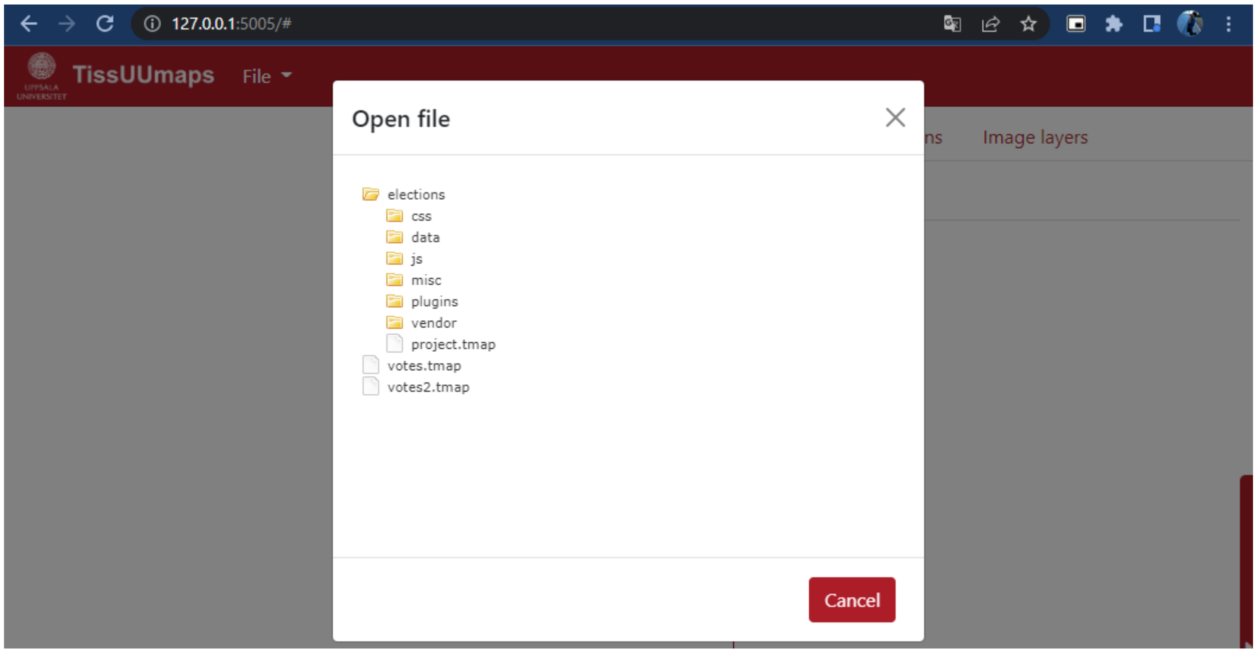
2.1.3 Load images using TissUUmaps server

If you are running TissUUmaps in server mode and not through the GUI, you must specify an image folder in the command line:

```
python -m tissuumaps "/home/username/Documents/myImages/" -p 5005
```

You can then access your images from your web browser by accessing the url http://localhost:5005, and using the `File > Open` menu.

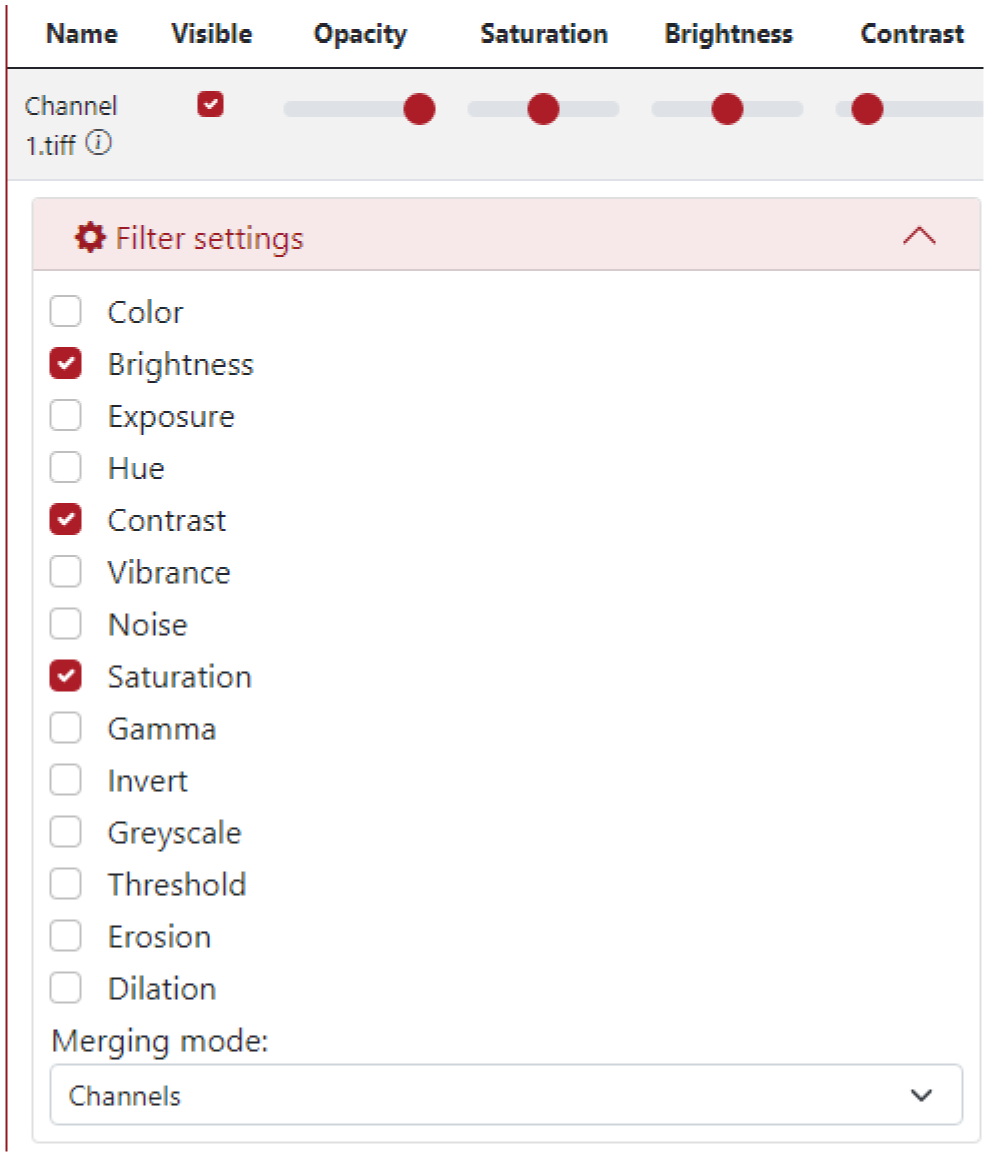




```
elections
    css
    data
    js
    misc
    plugins
    vendor
    project.tmap
votes.tmap
votes2.tmap
```

2.1.4 Apply filters

You can apply several filters to the images. The ones we can be adjusted by default are saturation, brightness, and contrast. Additionally, when opening the Filter settings menu, there are various other filters, such as exposure, noise, erosion, etc. When you check their box, they are automatically added to the filter panel above. The filter's sliders can be adjusted so that the filter is applied at the desired intensity. Another option in filter settings is merging mode (bottom part), where you can merge the channels as a composite.



Name	Visible	Opacity	Saturation	Brightness	Contrast
Channel 1.tiff ⓘ	✓	●	●	●	●

Filter settings

- [ ] Color
- [x] Brightness
- [ ] Exposure
- [ ] Hue
- [x] Contrast
- [ ] Vibrance
- [ ] Noise
- [x] Saturation
- [ ] Gamma
- [ ] Invert
- [ ] Greyscale
- [ ] Threshold
- [ ] Erosion
- [ ] Dilation

Merging mode: Channels

2.2 Markers

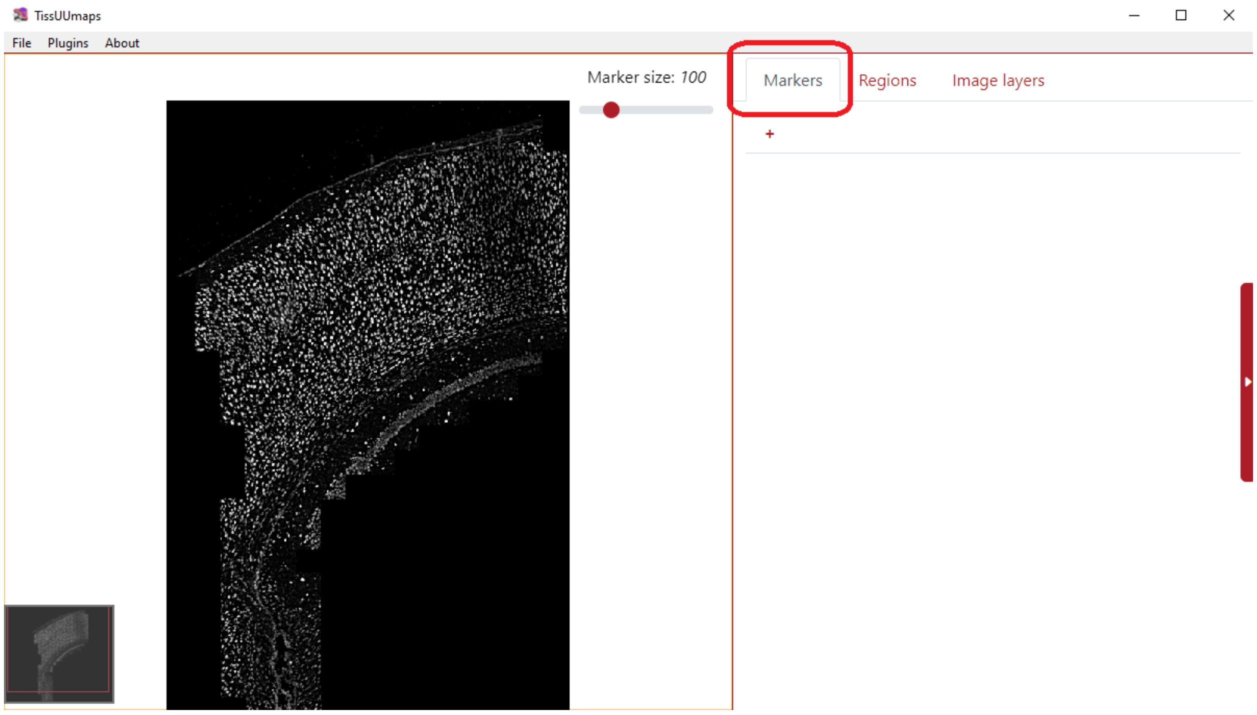
2.2.1 Supported marker format

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

You can load the markers when you select the *Markers* tab and click the button + as you can see in the figure below. You can click the plus several times to load various marker files.



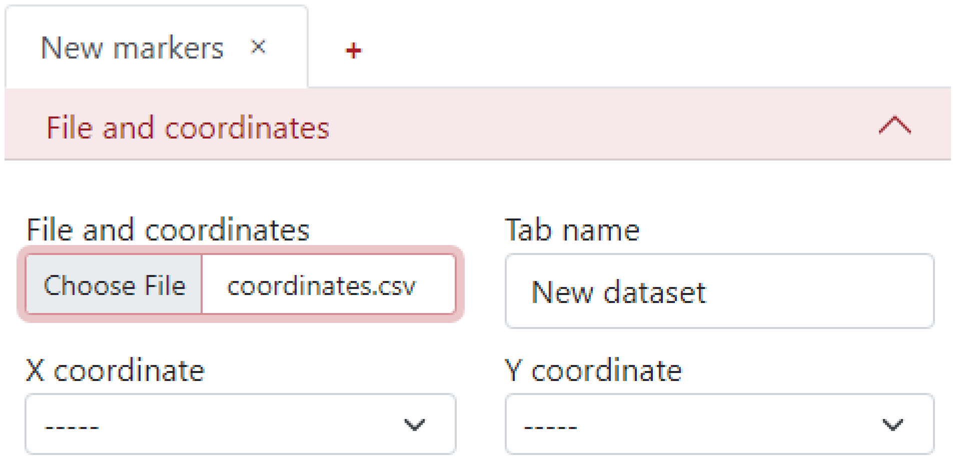
You can also load markers directly using drag and drop from a File Explorer if you are using the TissUUmaps GUI.

2.2.3 Markers settings

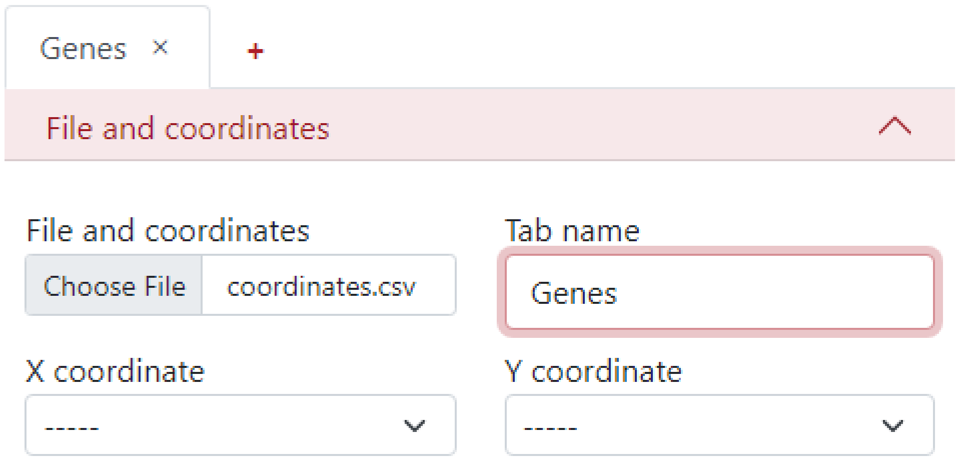
Before the markers are displayed you have to set up the markers settings.

File and coordinates

The first step is to select the desired file from your computer under the tab *File and coordinates - Choose file*.



You can change the *Tab name* to the desired name, so it is easier to navigate between them when there are more tabs.



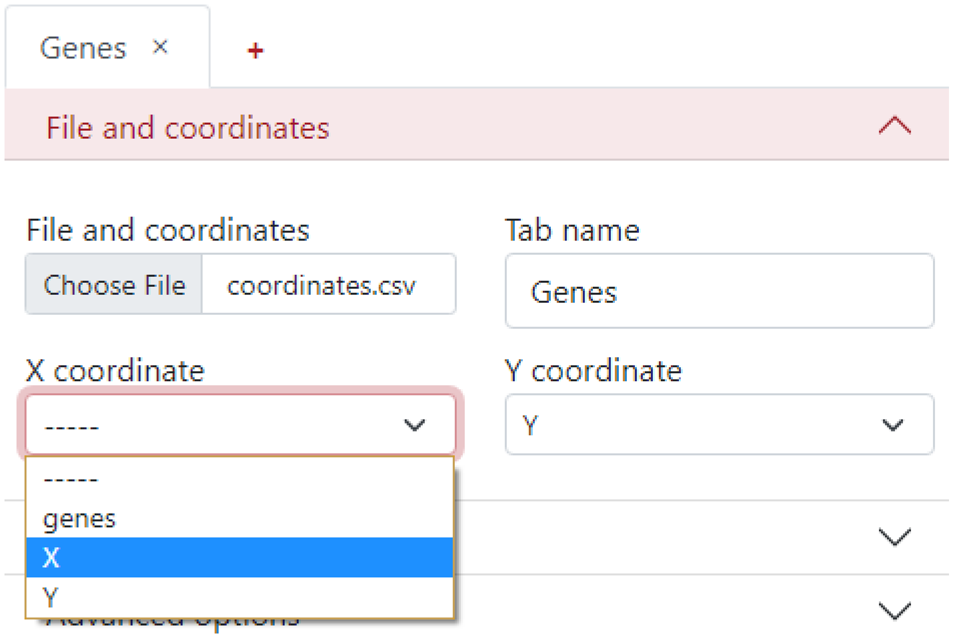
File and coordinates: Choose File coordinates.csv

Tab name: Genes

X coordinate: -----

Y coordinate: -----

The next step is to select the column names from the .csv file corresponding to the X and Y coordinates.



File and coordinates: Choose File coordinates.csv

Tab name: Genes

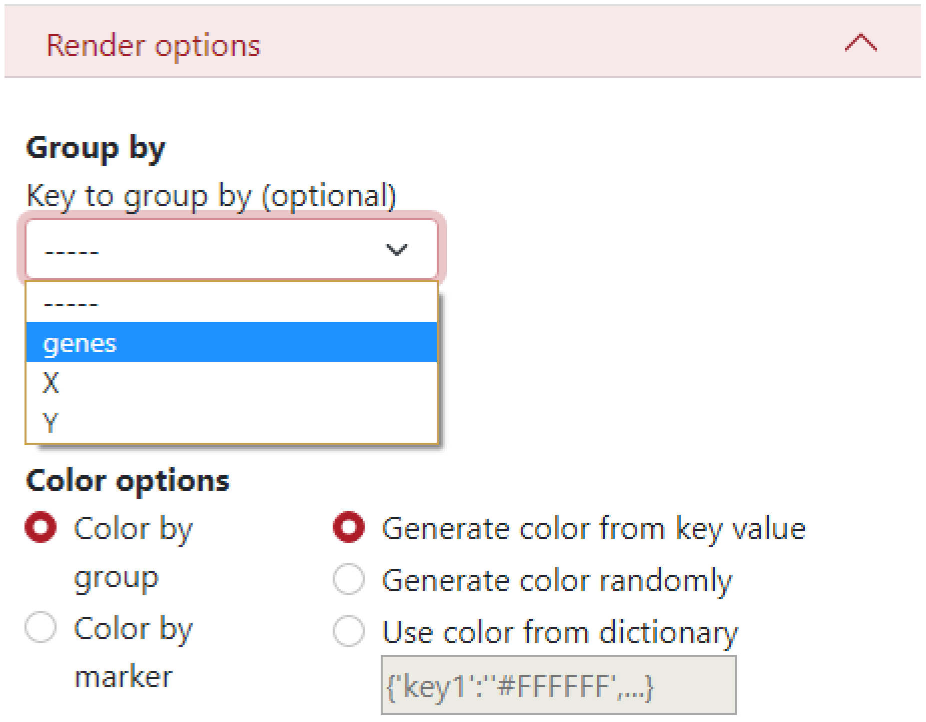
X coordinate: -----

-----
genes
X
Y

Y coordinate: Y

Render options

Here, you can define a *key to group by*, what is a column from the .csv file which will be used to display the dataset grouped by different colors and shapes of the markers. In this example, we use the column *genes*, where different colors and shapes of markers represent different genes.



Render options

Group by

Key to group by (optional)

- -----
- genes
- X
- Y

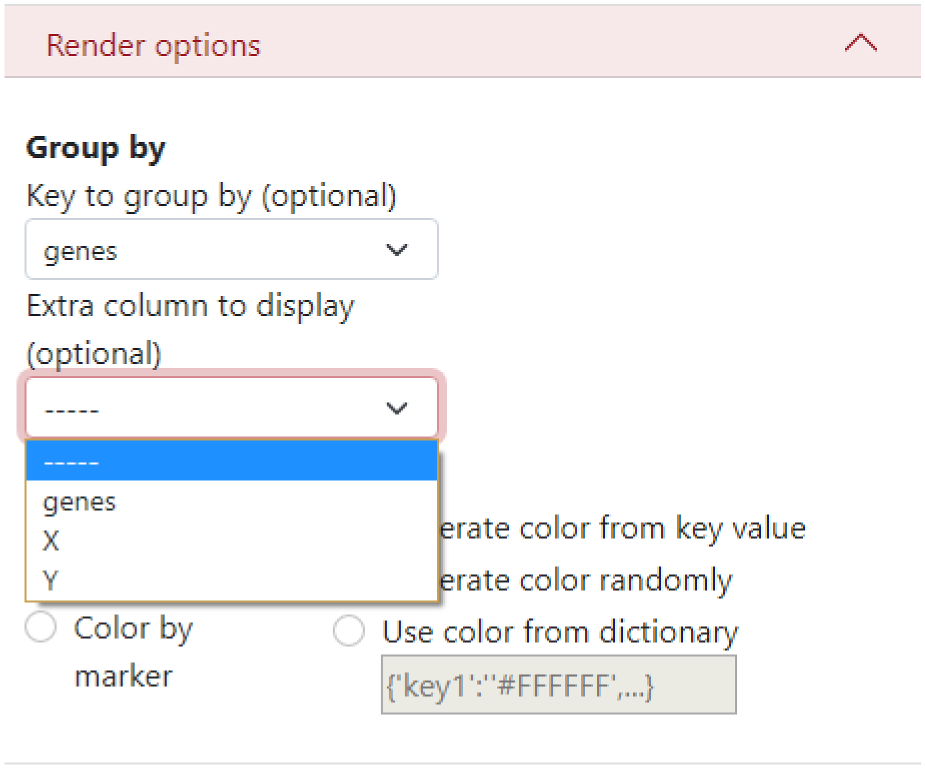
Color options

[✓ Color by group] Color by marker

[✓ Generate color from key value] Generate color randomly Use color from dictionary

{'key1':"#FFFFFF',...}

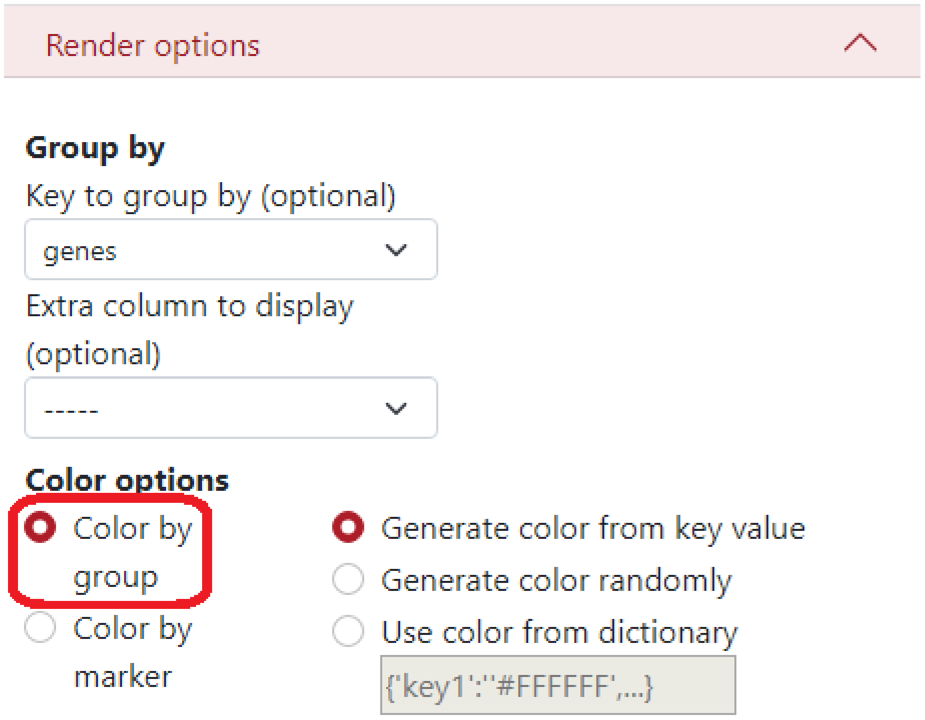
There is an option to display an extra column, for example when the data are clustered but you want to see the original genes and also the cluster names.


Render options
Group by
Key to group by (optional)
genes
Extra column to display (optional)

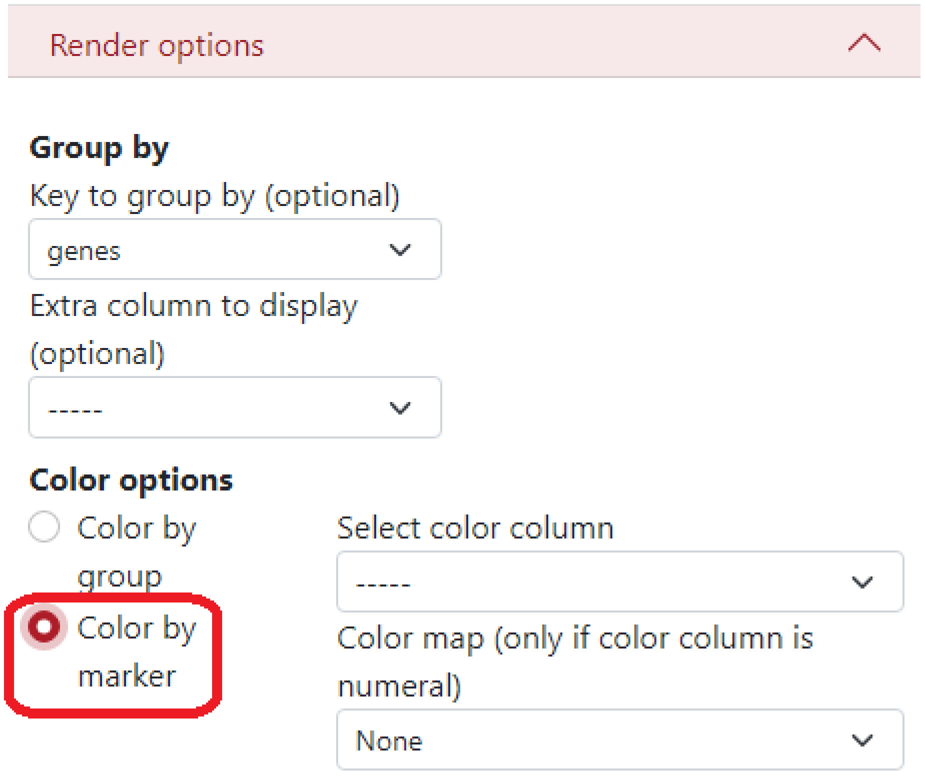
-----
genes
X
Y
erate color from key value
erate color randomly
Color by marker
Use color from dictionary
{'key1':'#FFFFFF',...}

In *Color options*, you can select to color by groups where each group has a different color. Then on the right side, you can select the color palette:

- Key value - Colors are generated from the name of the group (first 4 letters). Groups starting with the same letter have similar colors.
- Randomly - Colors are generated randomly.
- Dictionary - you can insert a specific dictionary in the text area which will be used for generating the colors.



If you want to *color by markers*, you have to select the column from the .csv file which will be used to create the colors, and the colormap, but only if the color column is numeral.

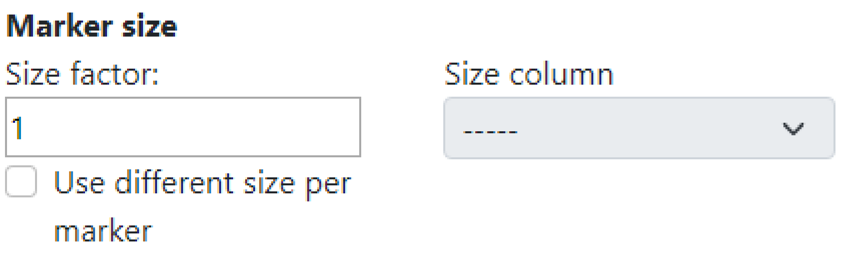

Render options
Group by
Key to group by (optional)
genes
Extra column to display (optional)

Color options
Color by group
Color by marker
Select color column

Color map (only if color column is numeral)
None

Advanced options

TissUUmaps tool contains also advanced options when working with the data. The first one is adjustable marker size. This is usually done in the right upper corner of the visualization panel. However, in the advanced setting, the user can change the size factor of the slider to any value.



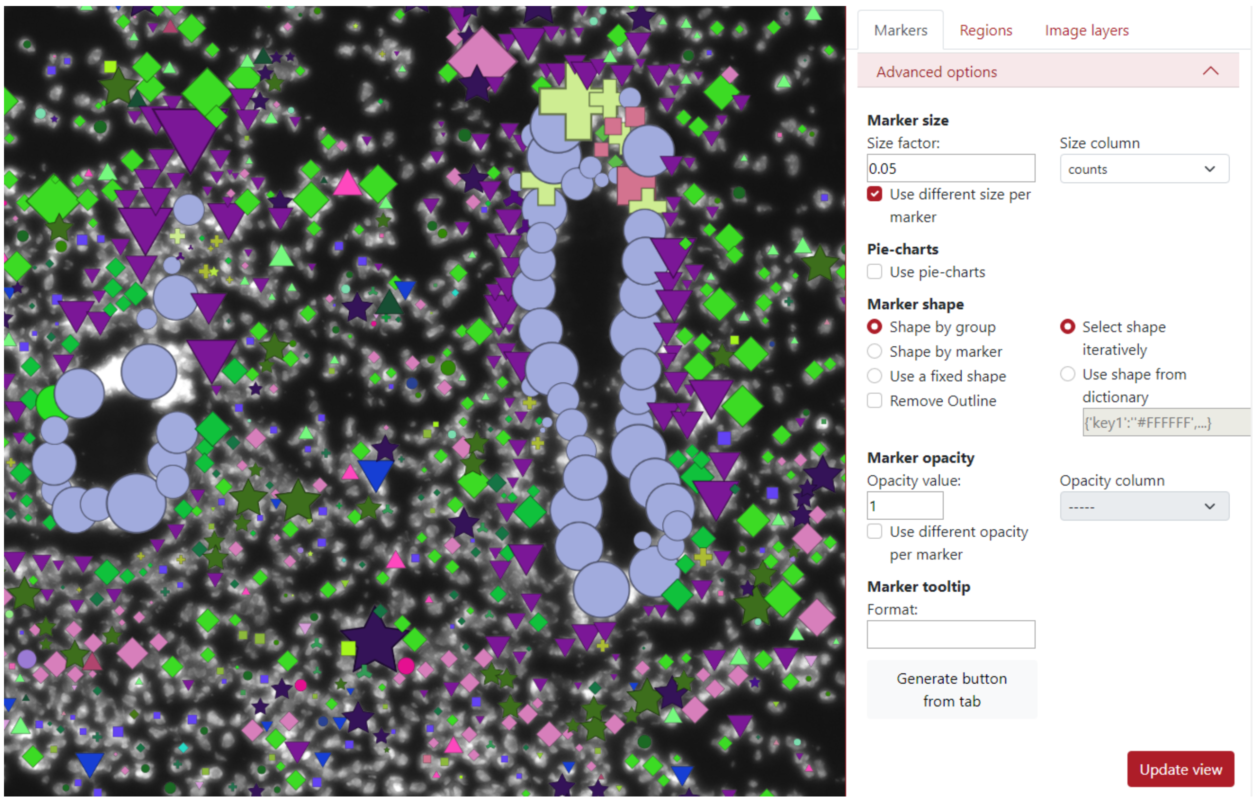
Marker size

Size factor: 1

Size column: -----

☐ Use different size per marker

Additionally, there can be used a different size per marker based on a selected column. In the example below, I chose column *counts* which represents the number of counts in that cell (marker). This means that a larger marker represents a cell that contains more counts in it.



Marker size

Size factor: 0.05

Size column: counts

☑ Use different size per marker

Pie-charts

☐ Use pie-charts

Marker shape

[✓ Shape by group] Shape by marker Use a fixed shape

☐ Remove Outline

[✓ Select shape iteratively] Use shape from dictionary {'key1':'#FFFFFF',...}

Marker opacity

Opacity value: 1

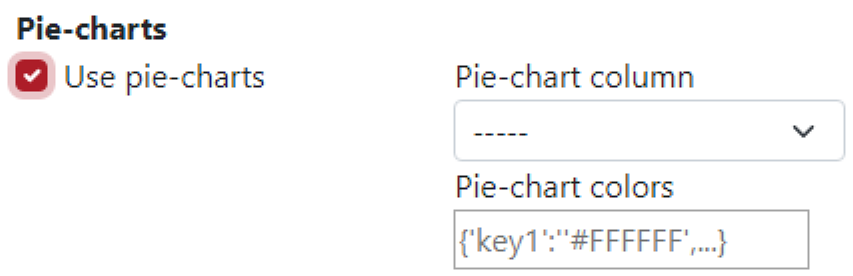
Opacity column: -----

☐ Use different opacity per marker

Marker tooltip

Format:

Another advanced option is the choice to display markers as pie-charts, it can represent the probability of that marker belonging to different groups. The user needs to select the *pie-chart column*, which contains mentioned probabilities for all the markers. All the probabilities for that specific marker need to be in a row divided by a semicolon.



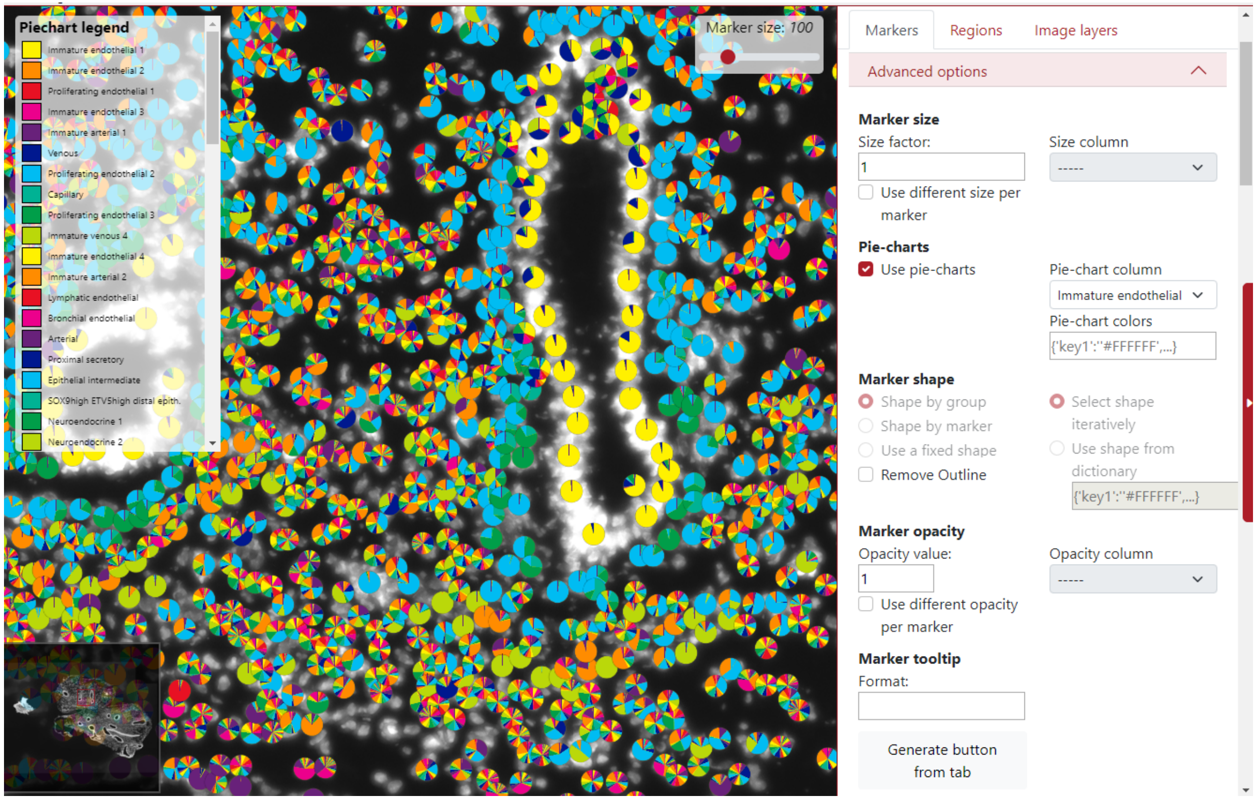
Pie-charts

[✓] Use pie-charts

Pie-chart column: -----

Pie-chart colors: {'key1':"#FFFFFF',...}

In the example below the pie-charts represent the probability of the marker being of each cell type. In the left upper corner can be seen the legend of the cell types. By default, there are only 10 colors so the colors are used in the loop. This can be changed by using pie-chart colors from a dictionary.



Marker size

Size factor: 1

Size column: -----

[ ] Use different size per marker

Pie-charts

[✓] Use pie-charts

Pie-chart column: Immature endothelial

Pie-chart colors: {'key1':"#FFFFFF',...}

Marker shape

[✓] Shape by group

( ) Shape by marker

( ) Use a fixed shape

[ ] Remove Outline

[✓] Select shape iteratively

( ) Use shape from dictionary

{'key1':"#FFFFFF',...}

Marker opacity

Opacity value: 1

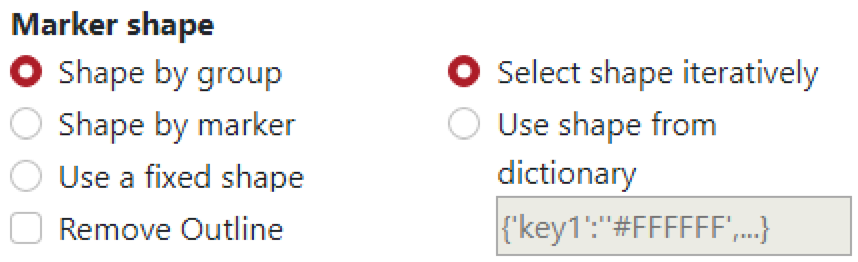
Opacity column: -----

[ ] Use different opacity per marker

Marker tooltip

Format:

The shape of the markers can also be changed. By default, it is set up to be selected by the group, which means that each group has a different marker shape chosen from the list of shapes iteratively. The user can also pre-define the shapes from the dictionary to ensure visualization robustness in different sessions.



Marker shape

[✓] Shape by group

( ) Shape by marker

( ) Use a fixed shape

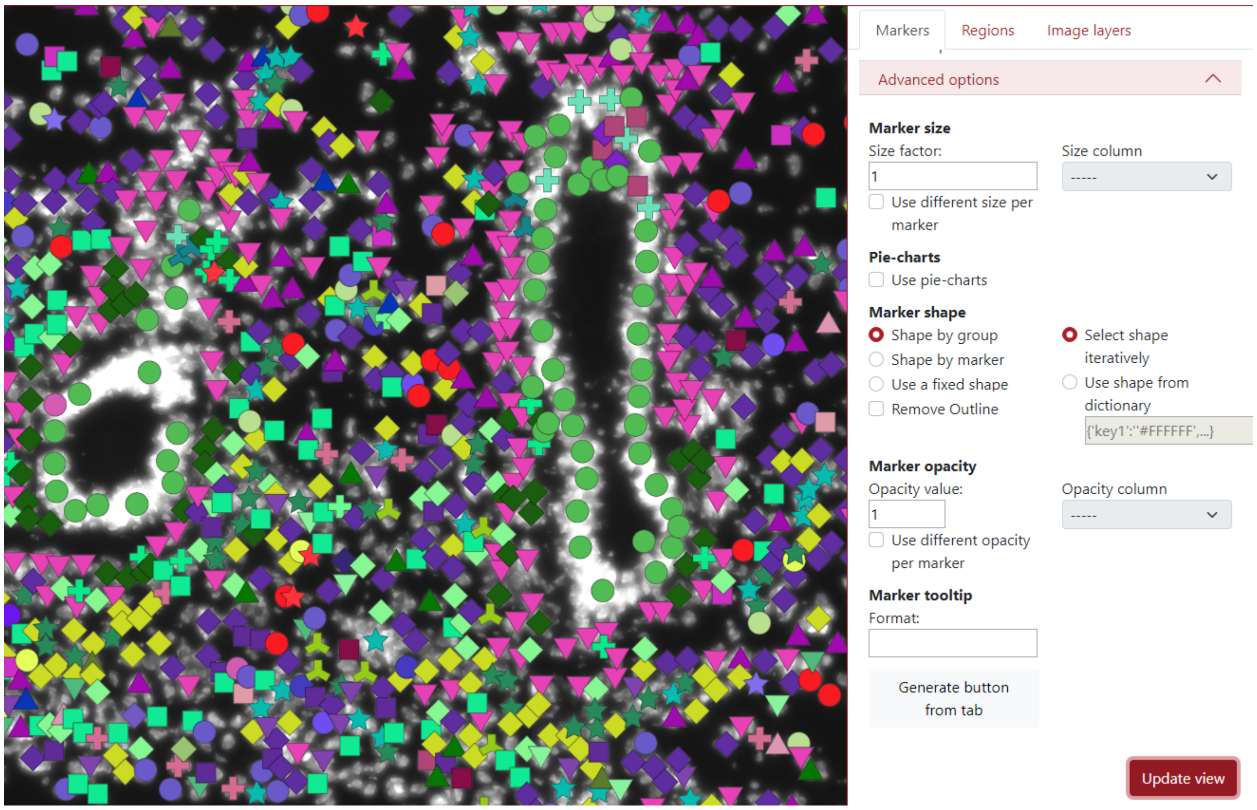
[ ] Remove Outline

[✓] Select shape iteratively

( ) Use shape from dictionary

{'key1':"#FFFFFF',...}

In the example below each group has specific color as well as a specific marker shape.



Markers | Regions | Image layers

Advanced options

Marker size

Size factor: 1

Size column: -----

Use different size per marker

Pie-charts

Use pie-charts

Marker shape

[✓ Shape by group]

Shape by marker

Use a fixed shape

Remove Outline

[✓ Select shape iteratively]

Use shape from dictionary

{'key1':"#FFFFFF',...}

Marker opacity

Opacity value: 1

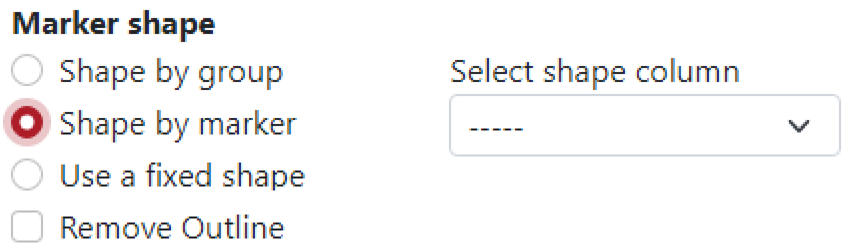
Opacity column: -----

Use different opacity per marker

Marker tooltip

Format:

Another option for the marker shape is *shape by marker*. Here, the user needs to select a column with category values, and each category is used for a different shape.



Marker shape

Shape by group

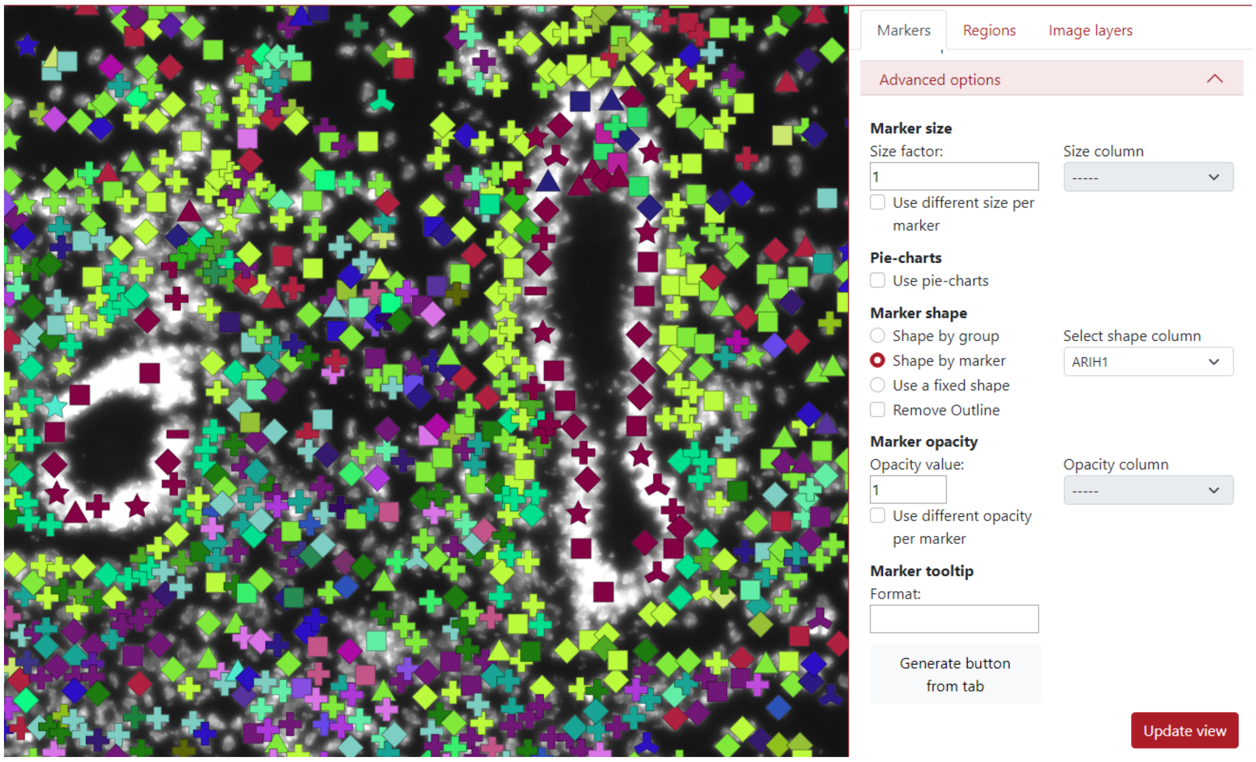
[✓ Shape by marker]

Use a fixed shape

Remove Outline

Select shape column: ------

In the example below, the selected column is ARIH1, which contains 10 categories, so you can see that there are 10 shapes in the visualization.



Marker size
Size factor: 1
Size column: -----
☐ Use different size per marker

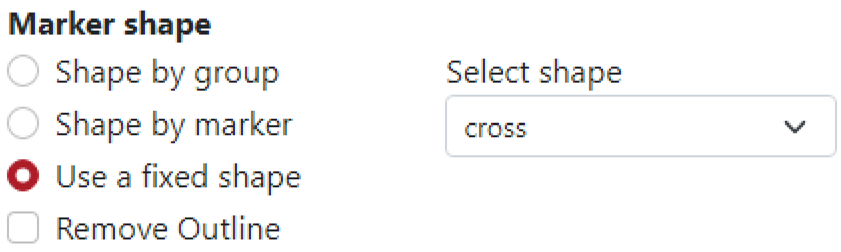
Pie-charts
☐ Use pie-charts

Marker shape
○ Shape by group
● Shape by marker
○ Use a fixed shape
☐ Remove Outline
Select shape column: ARIH1

Marker opacity
Opacity value: 1
Opacity column: -----
☐ Use different opacity per marker

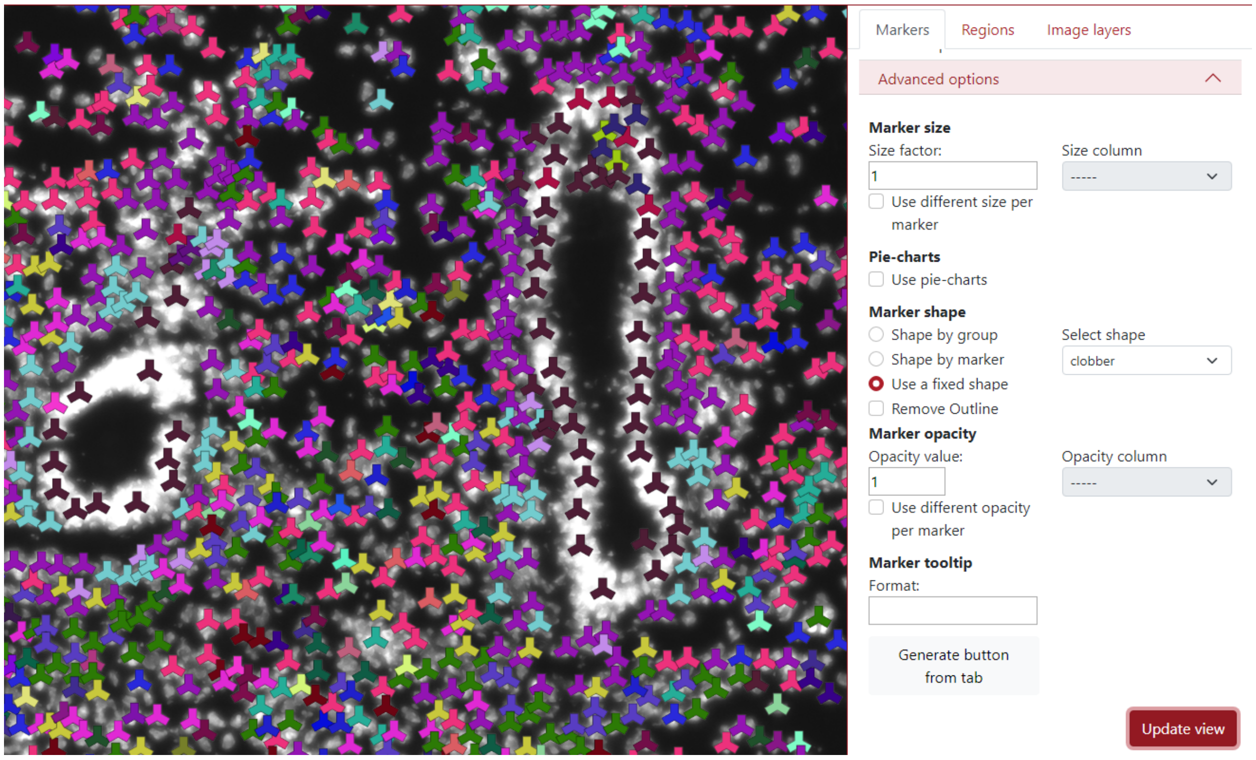
Marker tooltip
Format:

The third option in the marker shape is to *use a fixed shape*. This can be used if the user is not happy with all the different marker shapes and wants to make it homogeneous.



Marker shape
○ Shape by group
○ Shape by marker
● Use a fixed shape
☐ Remove Outline
Select shape: cross

In the example below, the selected shape is a clobber and you can see that all the markers are in the shape of a clobber.



Marker size
Size factor: 1
Size column: -----
☐ Use different size per marker

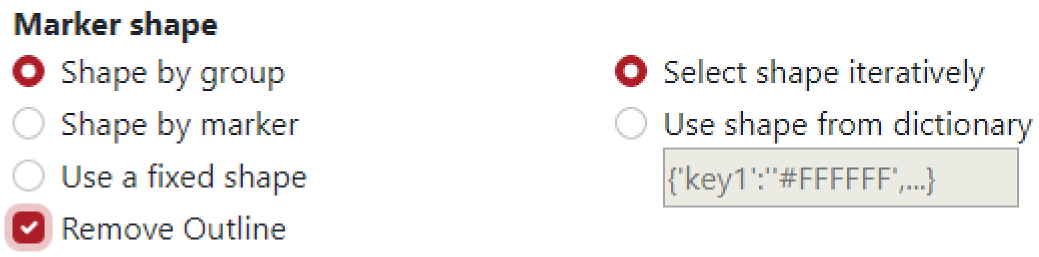
Pie-charts
☐ Use pie-charts

Marker shape
○ Shape by group
○ Shape by marker
● Use a fixed shape
☐ Remove Outline
Select shape: clobber

Marker opacity
Opacity value: 1
Opacity column: -----
☐ Use different opacity per marker

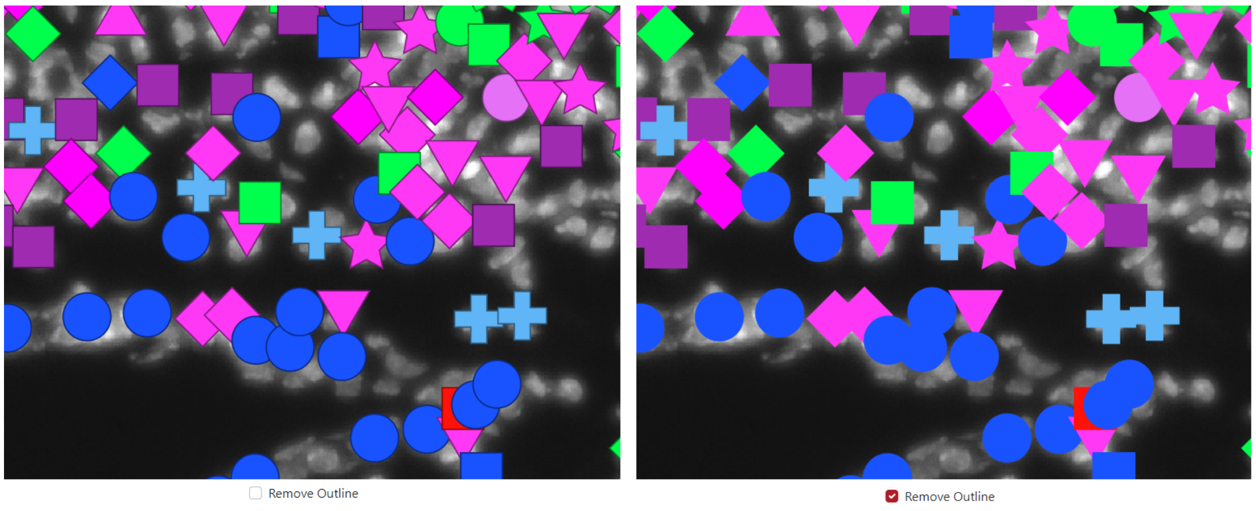
Marker tooltip
Format:

The last option in the marker shape is to *remove outline*. This can be used to remove the dark outline of markers when the check box is checked.

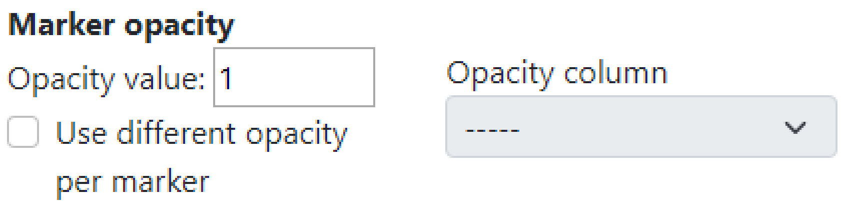


Marker shape
● Shape by group
○ Shape by marker
○ Use a fixed shape
☑ Remove Outline
● Select shape iteratively
○ Use shape from dictionary
{'key1':"#FFFFFF',...}

In the example below, the outline is included on the left side and the outline is removed on the right side.



The next advanced option is *marker opacity* which is adjustable. The user can change the opacity in order to display things underneath.



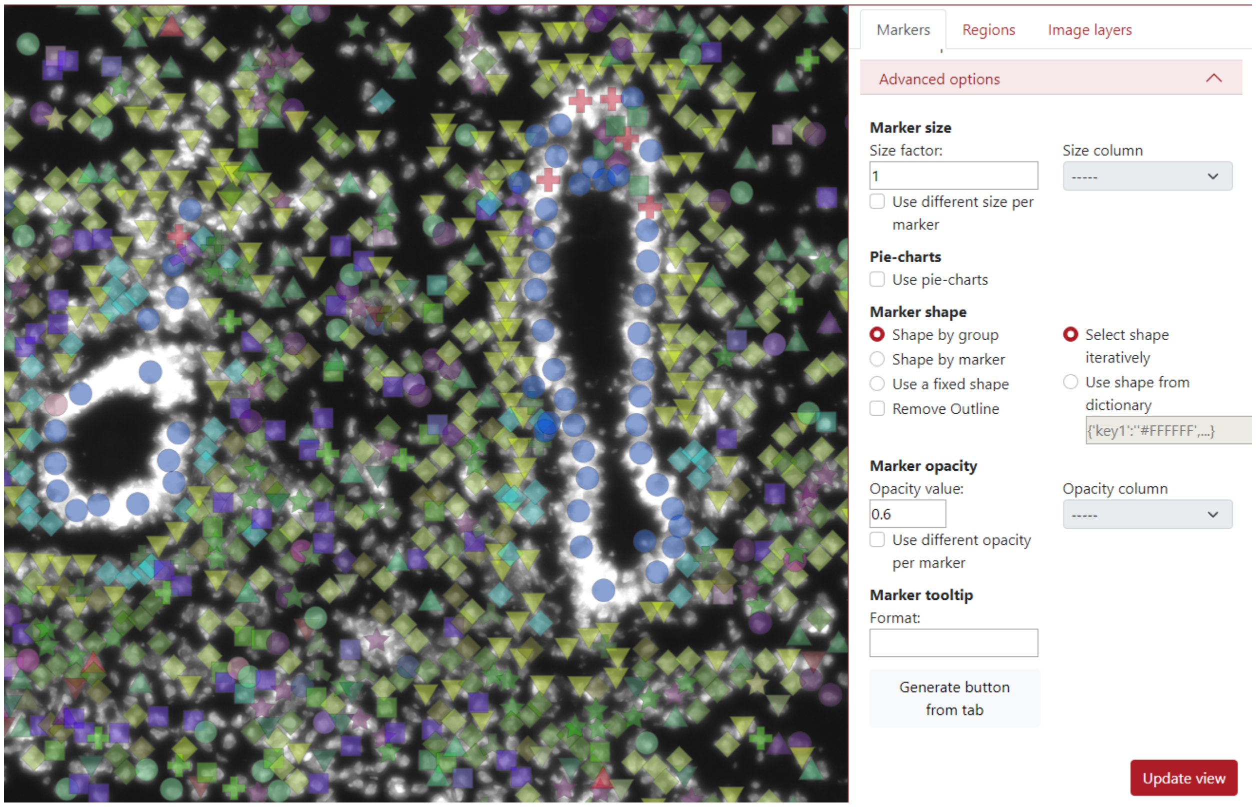
Marker opacity

Opacity value: 1

Use different opacity per marker

Opacity column: -----

In the example below, the opacity value was set to 0.6 which made the markers a bit transparent.



Marker size

Size factor: 1

Size column: -----

Use different size per marker

Pie-charts

Use pie-charts

Marker shape

[✓ Shape by group]

Shape by marker

Use a fixed shape

Remove Outline

[✓ Select shape iteratively]

Use shape from dictionary

{'key1':"#FFFFFF',...}

Marker opacity

Opacity value: 0.6

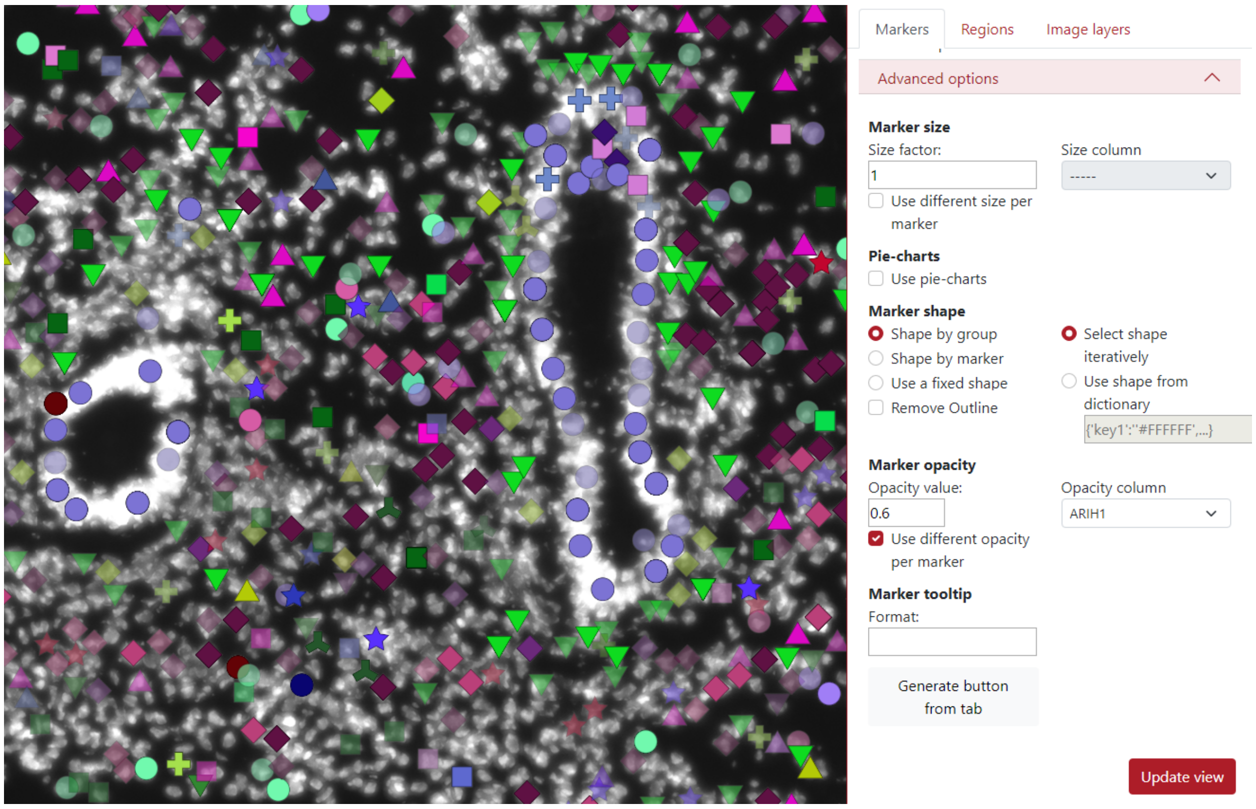
Opacity column: -----

Use different opacity per marker

Marker tooltip

Format:

In the example below, we checked *use different opacity per marker*. The user needs to select opacity column which will be used for displaying different opacities in markers.



Marker size
Size factor: 1
Size column: -----
☐ Use different size per marker

Pie-charts
☐ Use pie-charts

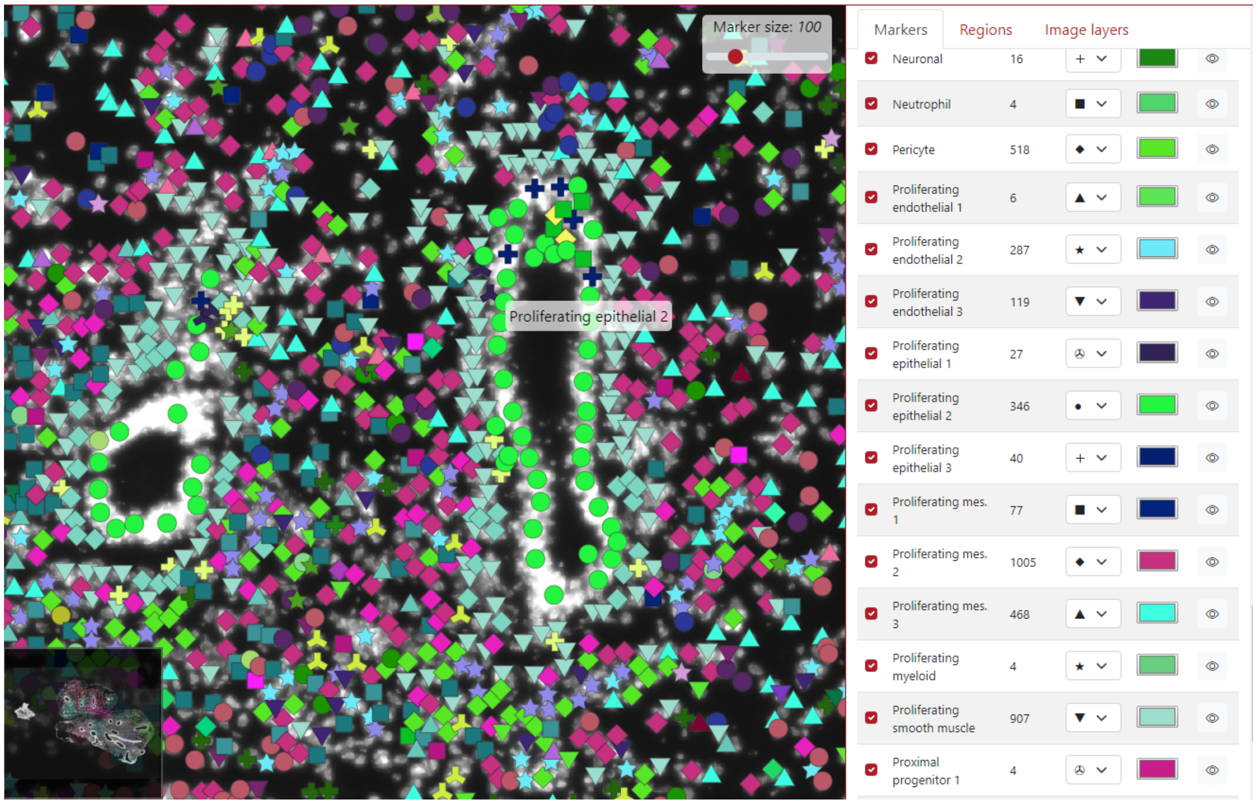
Marker shape
[✓ Shape by group]
Shape by marker
Use a fixed shape
☐ Remove Outline
[✓ Select shape iteratively]
Use shape from dictionary
{'key1':"#FFFFFF',...}

Marker opacity
Opacity value: 0.6
Opacity column: ARIH1
☑ Use different opacity per marker

Marker tooltip
Format:

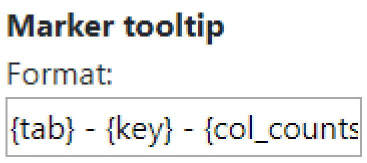
The next option is the *marker tooltip*, which is the text which is displayed when the user clicks on the marker. By default, it displayed the key group the marker belongs to.

As you can see in the example below, the green marker we clicked on belongs to the cell type Proliferating epithelial 2.



✓	Neuronal	16	+	🟩	👁
✓	Neutrophil	4	■	🟩	👁
✓	Pericyte	518	◆	🟩	👁
✓	Proliferating endothelial 1	6	▲	🟩	👁
✓	Proliferating endothelial 2	287	★	🟦	👁
✓	Proliferating endothelial 3	119	▼	🟪	👁
✓	Proliferating epithelial 1	27	☮	🟪	👁
✓	Proliferating epithelial 2	346	●	🟩	👁
✓	Proliferating epithelial 3	40	+	🟦	👁
✓	Proliferating mes. 1	77	■	🟦	👁
✓	Proliferating mes. 2	1005	◆	🟥	👁
✓	Proliferating mes. 3	468	▲	🟦	👁
✓	Proliferating myeloid	4	★	🟩	👁
✓	Proliferating smooth muscle	907	▼	🟩	👁
✓	Proximal progenitor 1	4	☮	🟪	👁

However, this can be modified by writing text into the text area. In this example, we wrote {tab} - {key} - {col_counts}. *{tab}* represents the tab name that we set when we loaded the markers, *{key}* represents the key group to which the marker belongs and *{col_counts}* represents the value of that marker in the column called counts. The word counts can be replaced by any column name in order to display it on the tooltip.

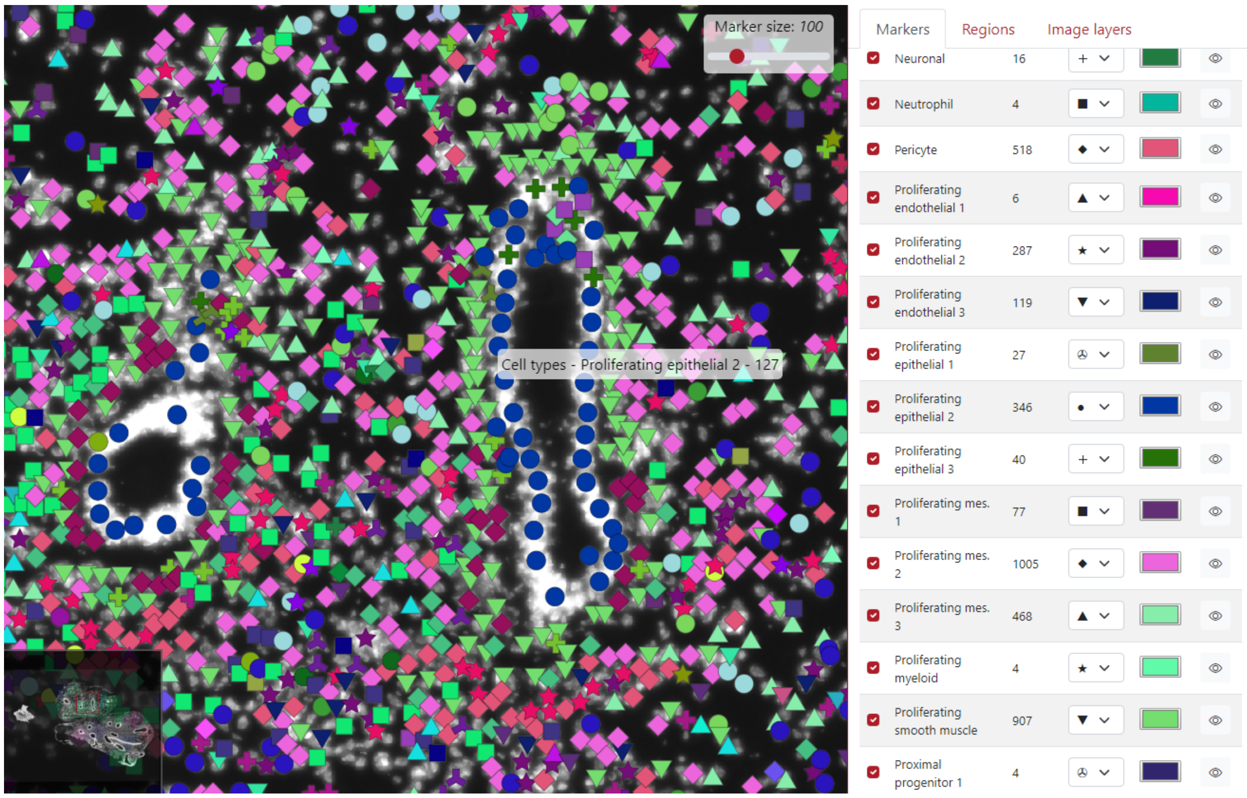


Marker tooltip

Format:

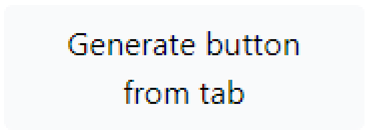
{tab} - {key} - {col_counts

In the example below, the green dot which we clicked on is from the tab Cell types, belongs to the cell type Proliferating epithelial 2 and it has 127 counts in it.

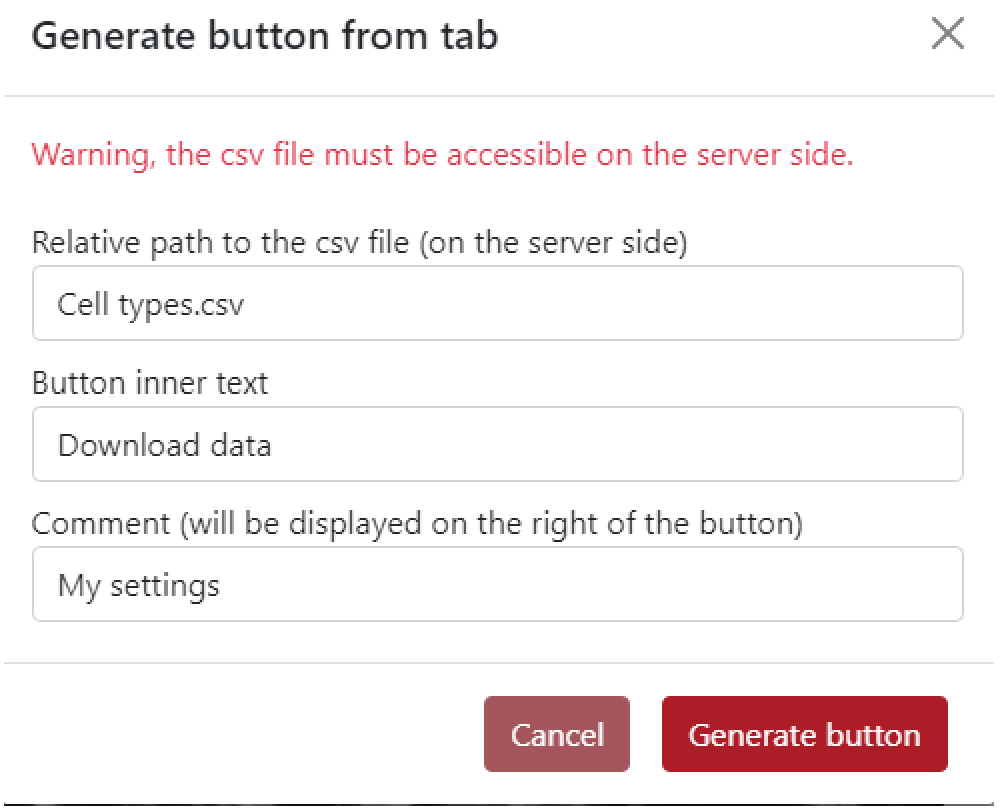


✓	Neuronal	16	+	🟩
✓	Neutrophil	4	■	🟩
✓	Pericyte	518	◆	🟥
✓	Proliferating endothelial 1	6	▲	🟪
✓	Proliferating endothelial 2	287	★	🟪
✓	Proliferating endothelial 3	119	▼	🟦
✓	Proliferating epithelial 1	27	✇	🟩
✓	Proliferating epithelial 2	346	●	🟦
✓	Proliferating epithelial 3	40	+	🟩
✓	Proliferating mes. 1	77	■	🟪
✓	Proliferating mes. 2	1005	◆	🟪
✓	Proliferating mes. 3	468	▲	🟩
✓	Proliferating myeloid	4	★	🟩
✓	Proliferating smooth muscle	907	▼	🟩
✓	Proximal progenitor 1	4	✇	🟦

The last advanced option is the button *Generate button from tab*. This button incorporates all the display settings the user set up into a single button.



The user can choose the relative path to the csv file, button inner text and comment which will be displayed next to the button.



Generate button from tab

Warning, the csv file must be accessible on the server side.

Relative path to the csv file (on the server side)

Cell types.csv

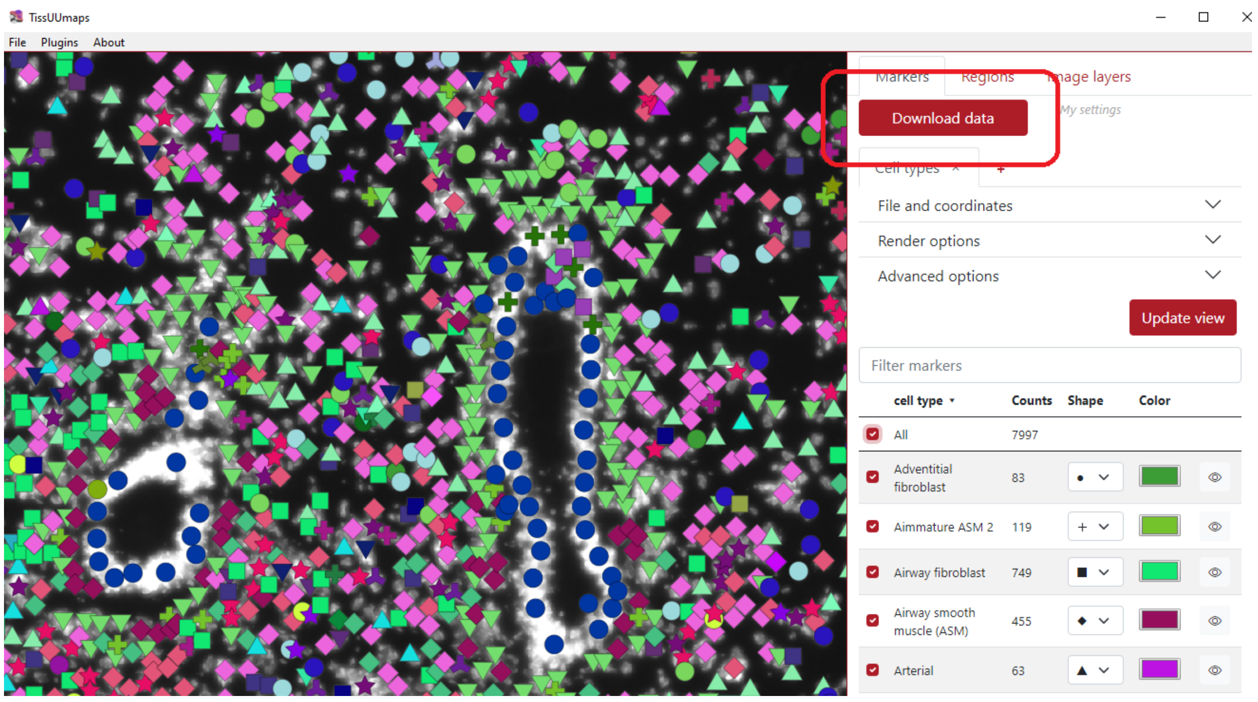
Button inner text

Download data

Comment (will be displayed on the right of the button)

My settings

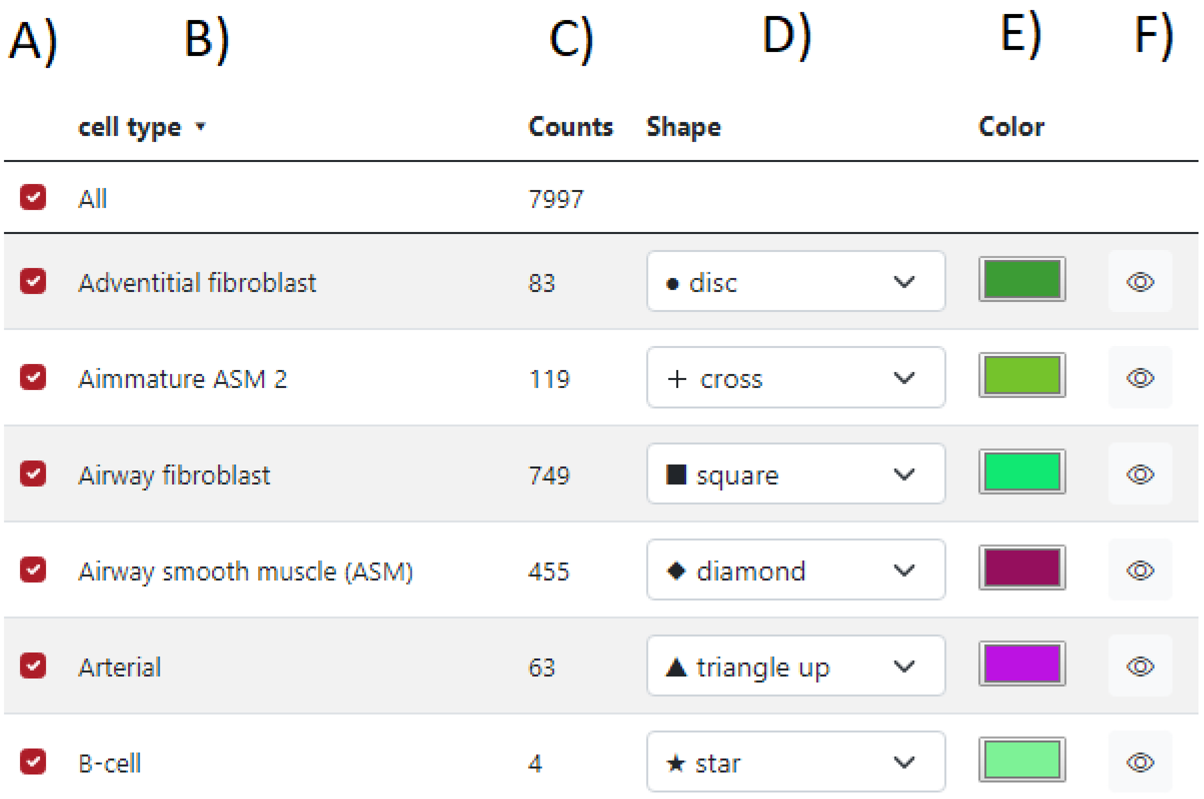
In the example below, you can see the generated button *Download data* placed on the top of the tabs panel. On the right of the button is placed text *My settings*.



cell type	Counts	Shape	Color
All	7997		
Adventitial fibroblast	83	•	🟩
Aimmature ASM 2	119	+	🟩
Airway fibroblast	749	■	🟩
Airway smooth muscle (ASM)	455	◆	🟥
Arterial	63	▲	🟪

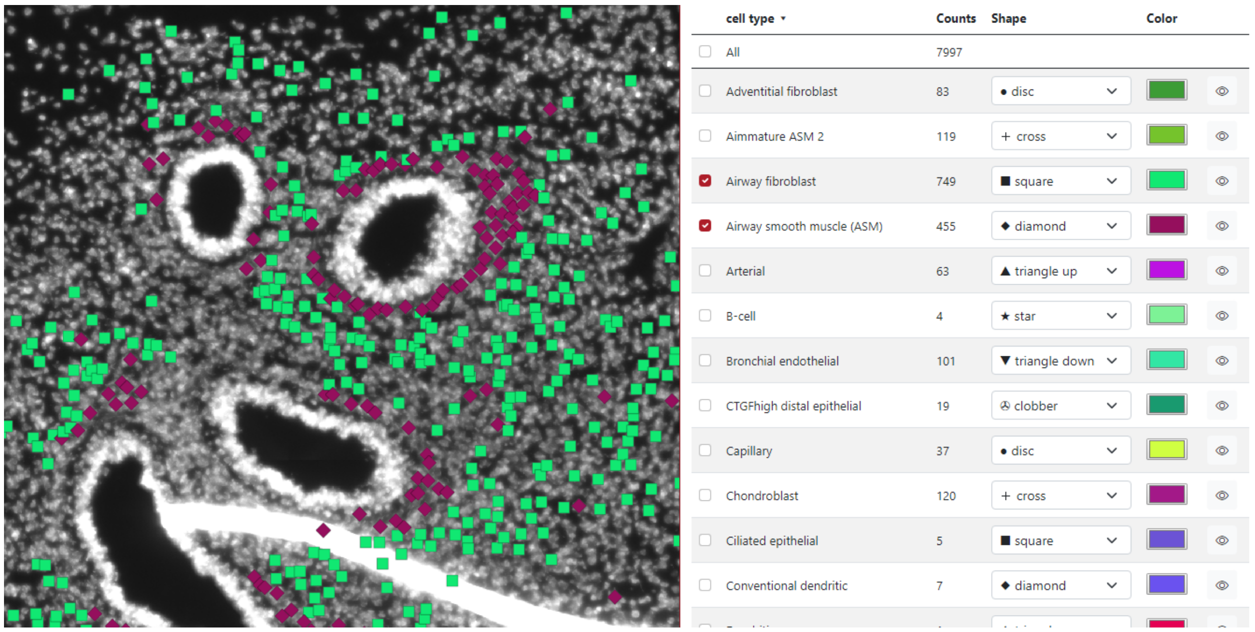
Table of markers

When the markers are loaded, a table of markers will appear in order to interact with the marker. Each row represents a group of markers with a specific color and shape. In the figure below, column *A)* represents if a specific row of markers is displayed or not, the second column *B)* represents the list of groups, the third column *C)* represents group counts, the fourth column *D)* represents the shape of the group markers, the fifth column *E)* represents the color of the group markers and the sixth column *F)* can display specific group when the cursor is on the eye icon.



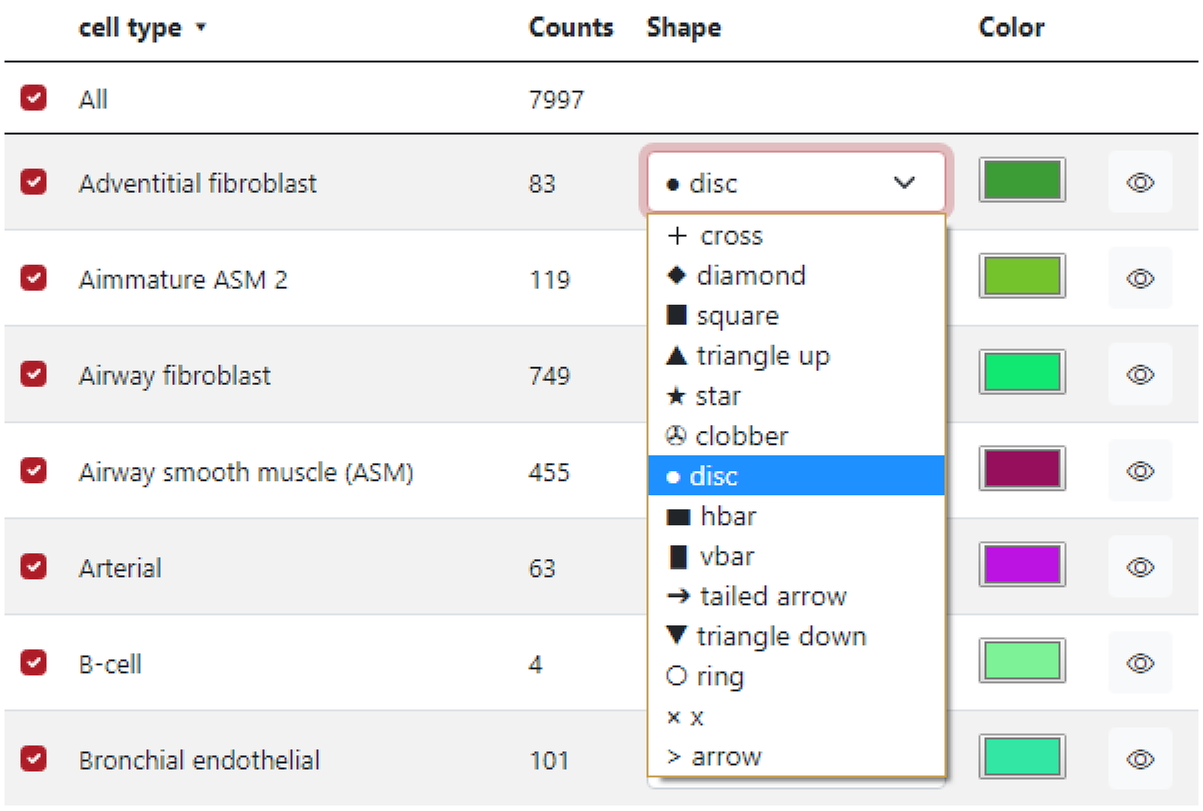
	cell type ▾	Counts	Shape	Color	
✓	All	7997			
✓	Adventitial fibroblast	83	● disc	🟩	👁
✓	Aimmature ASM 2	119	+ cross	🟩	👁
✓	Airway fibroblast	749	■ square	🟩	👁
✓	Airway smooth muscle (ASM)	455	◆ diamond	🟥	👁
✓	Arterial	63	▲ triangle up	🟪	👁
✓	B-cell	4	★ star	🟩	👁

If the check box is checked - the group is displayed, if the check box is unchecked - the group is not displayed. In the example below, we checked two groups of cell types: Airway Fibroblast and Airway smooth muscle, and only these two groups are displayed on the left visualization panel. The first checkbox *All* ensures displaying of all the markers.



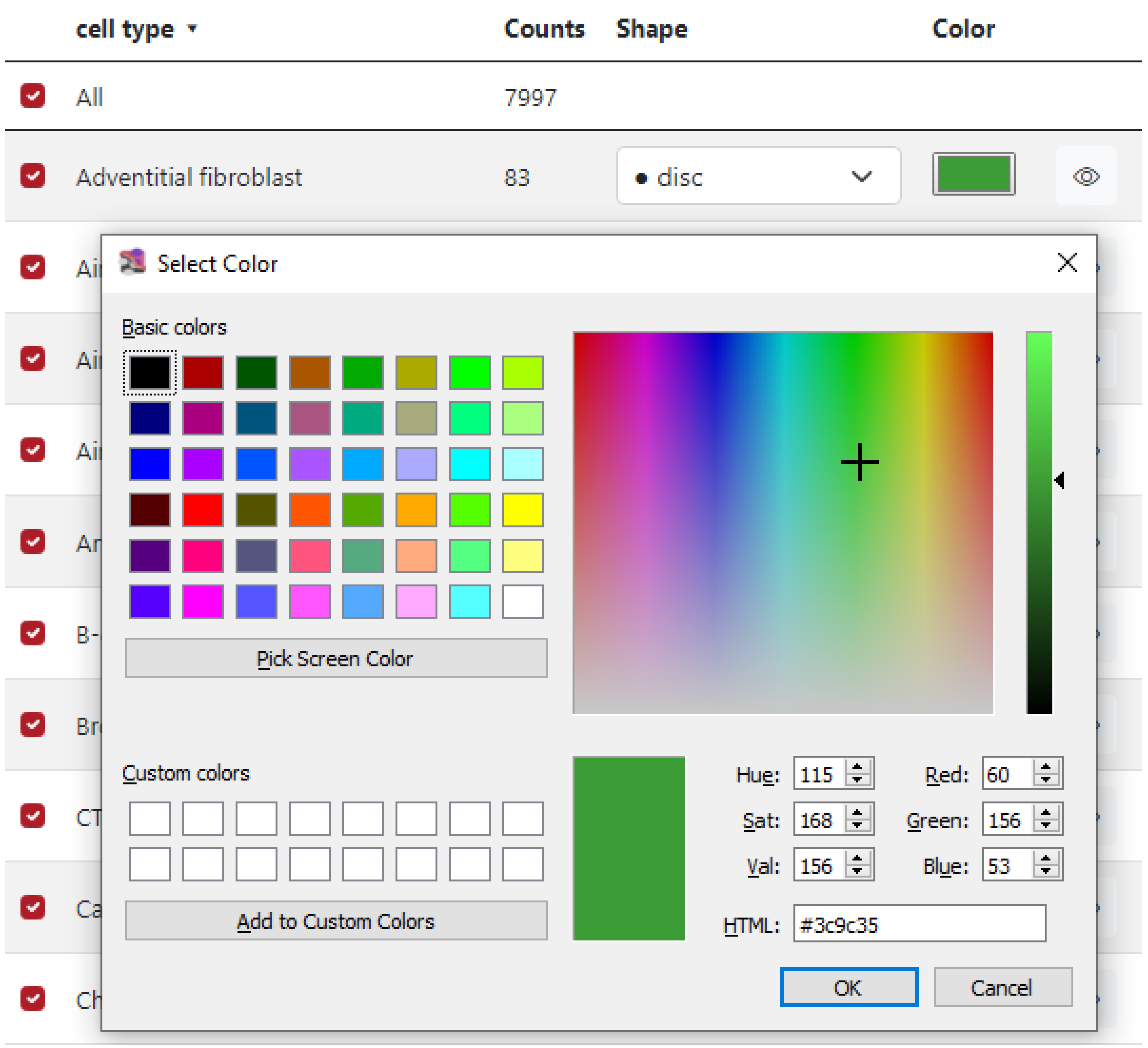
	cell type ▾	Counts	Shape	Color	
☐	All	7997			
☐	Adventitial fibroblast	83	● disc	🟩	👁
☐	Aimmature ASM 2	119	+ cross	🟩	👁
✓	Airway fibroblast	749	■ square	🟩	👁
✓	Airway smooth muscle (ASM)	455	◆ diamond	🟥	👁
☐	Arterial	63	▲ triangle up	🟪	👁
☐	B-cell	4	★ star	🟩	👁
☐	Bronchial endothelial	101	▼ triangle down	🟩	👁
☐	CTGFhigh distal epithelial	19	✇ clobber	🟩	👁
☐	Capillary	37	● disc	🟨	👁
☐	Chondroblast	120	+ cross	🟪	👁
☐	Ciliated epithelial	5	■ square	🟦	👁
☐	Conventional dendritic	7	◆ diamond	🟦	👁

In the fourth column *Shape*, the user can select which shape is preferred for each marker group. In the figure below, there is a list of 14 different shapes which can be used.



	cell type ▾	Counts	Shape	Color	
✓	All	7997			
✓	Adventitial fibroblast	83	● disc	🟩	👁
✓	Aimmature ASM 2	119		🟩	👁
✓	Airway fibroblast	749		🟩	👁
✓	Airway smooth muscle (ASM)	455		🟥	👁
✓	Arterial	63		🟪	👁
✓	B-cell	4		🟩	👁
✓	Bronchial endothelial	101		🟩	👁

In the fifth column *Color*, the user can select which color is preferred for each marker group. In the figure below, it is possible to choose from some list of basic colors, select a specific color by the cursor from the palette and also use numbers to generate color, either RGB, HSV, or HTML.

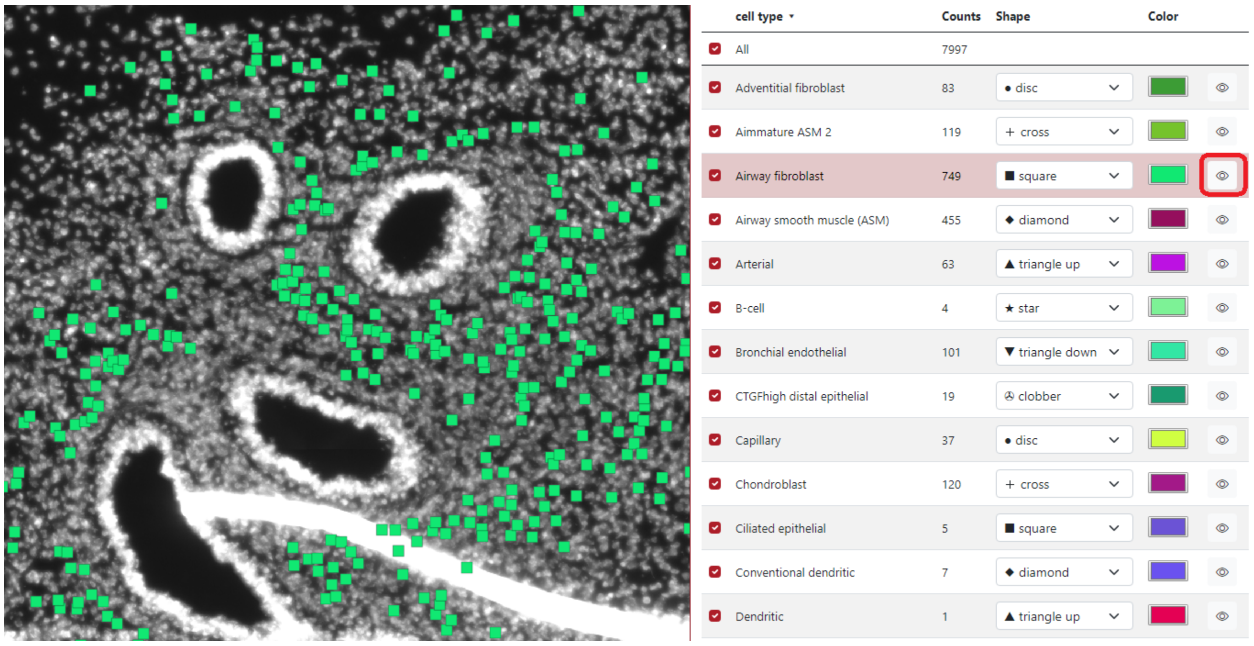


	cell type ▾	Counts	Shape	Color	
✓	All	7997			
✓	Adventitial fibroblast	83	● disc	🟩	👁

Select Color

Hue: 115
Sat: 168
Val: 156
Red: 60
Green: 156
Blue: 53
HTML: #3c9c35

In the example below can be seen that if the cursor is placed on the eye icon in the row Airway fibroblast, only markers of this group are displayed on the visualization panel.



	cell type ▾	Counts	Shape	Color	
✓	All	7997			
✓	Adventitial fibroblast	83	● disc	🟩	👁
✓	Aimmature ASM 2	119	+ cross	🟩	👁
✓	Airway fibroblast	749	■ square	🟩	👁
✓	Airway smooth muscle (ASM)	455	◆ diamond	🟪	👁
✓	Arterial	63	▲ triangle up	🟪	👁
✓	B-cell	4	★ star	🟩	👁
✓	Bronchial endothelial	101	▼ triangle down	🟩	👁
✓	CTGFhigh distal epithelial	19	✇ clobber	🟩	👁
✓	Capillary	37	● disc	🟨	👁
✓	Chondroblast	120	+ cross	🟪	👁
✓	Ciliated epithelial	5	■ square	🟪	👁
✓	Conventional dendritic	7	◆ diamond	🟪	👁
✓	Dendritic	1	▲ triangle up	🟥	👁

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:

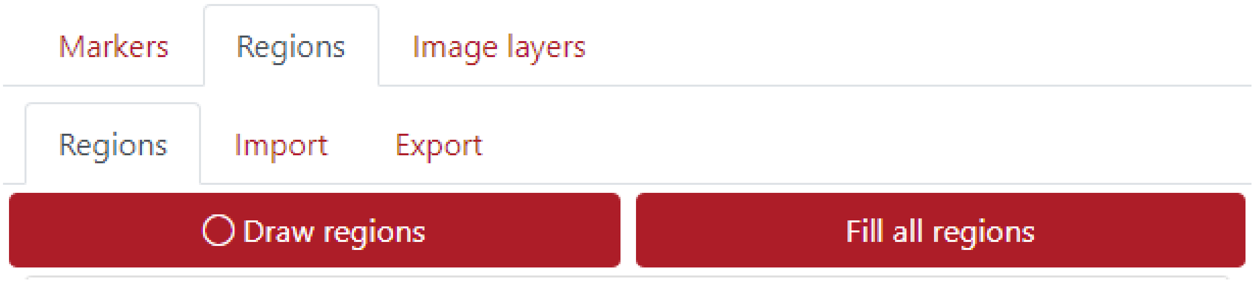
- Polygon
- Multipolygon

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

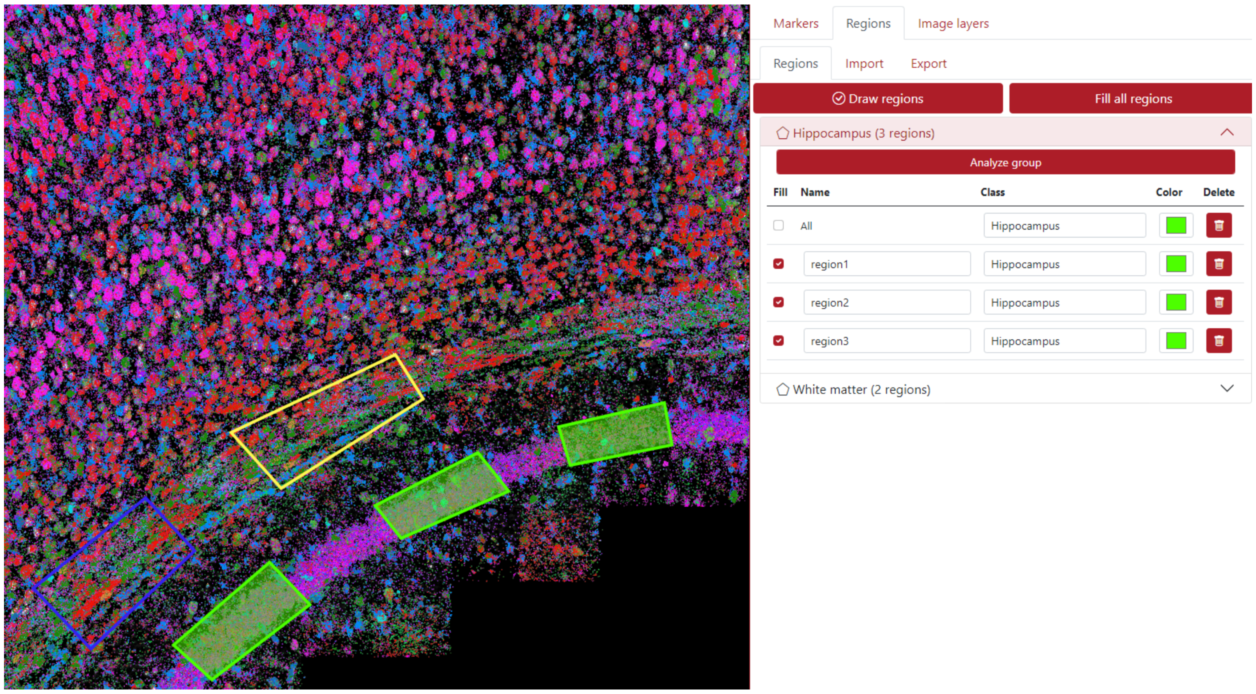
2.3.2 Draw Regions

Regions are polygons that can be drawn by the user or imported from an external file. If the user clicks on the button *Draw regions* shown underneath, the button is checked and the user can draw an unlimited number of regions. The user clicks on the image to outline the region of interest, then click on the first point to close the region and the region will appear in the right panel.

The button *Fill all regions* fills the inner part of all the regions by semitransparent color. This can be done separately by clicking on the check box next to individual regions.

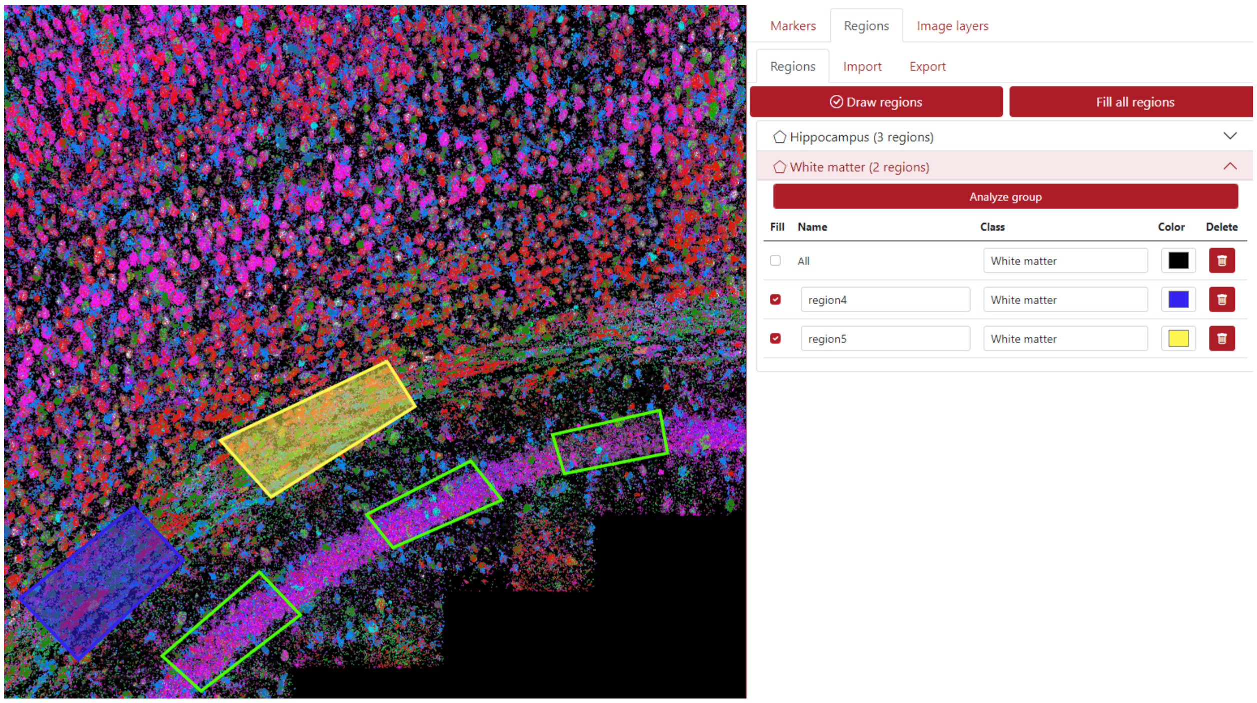


In the example below are three drawn regions selecting the hippocampus areas, all set to green color. You can set the class name in the *Class* column. You can see that the drawn regions are categorized into two main groups Hippocampus and White matter. The user can create an unlimited number of groups depending on his interest. This helps to have the regions organized and also it is very useful when exporting regions.



Fill	Name	Class	Color	Delete
☐	All	Hippocampus	🟩	🗑
✓	region1	Hippocampus	🟩	🗑
✓	region2	Hippocampus	🟩	🗑
✓	region3	Hippocampus	🟩	🗑

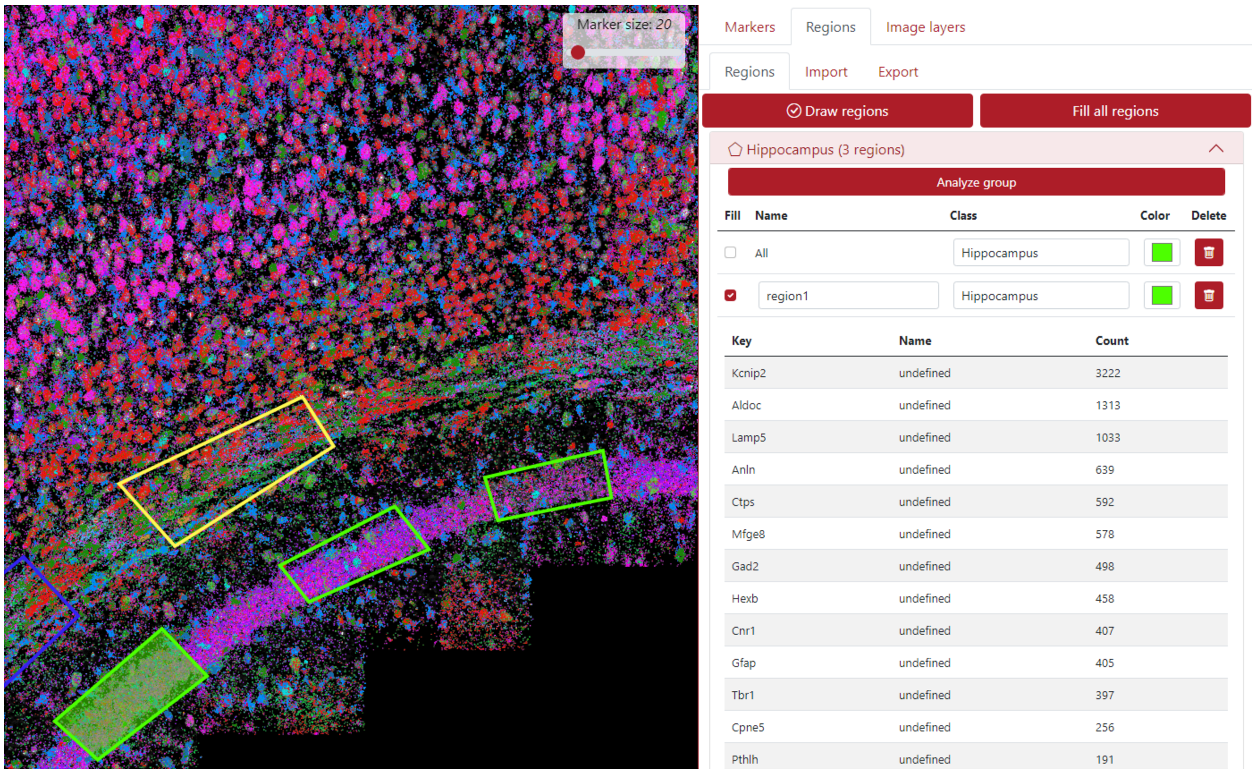
In the example below are two drawn regions selecting the white matter areas, one is set to yellow and the other one to blue color. The regions are interchangeable between groups, so if you want to move a region from group Hippocampus to group White matter, just change the class name in the region's row and it will be automatically moved to the desired group.



Fill	Name	Class	Color	Delete
☐	All	White matter	■	🗑
☑	region4	White matter	🟦	🗑
☑	region5	White matter	🟨	🗑

2.3.3 Analyze Regions

The regions can be analyzed, meaning displaying a list of all the marker keys with their counts inside that region (expression). The example below shows the analyzed region1. In this case, the analysis contains gene expression, so the column *Key* contains a list of genes, and column *Name* could show an additional column from the dataset, in this case, it is undefined since we provided only column Genes. The last column *Count* shows the number of each gene inside the analyzed region.

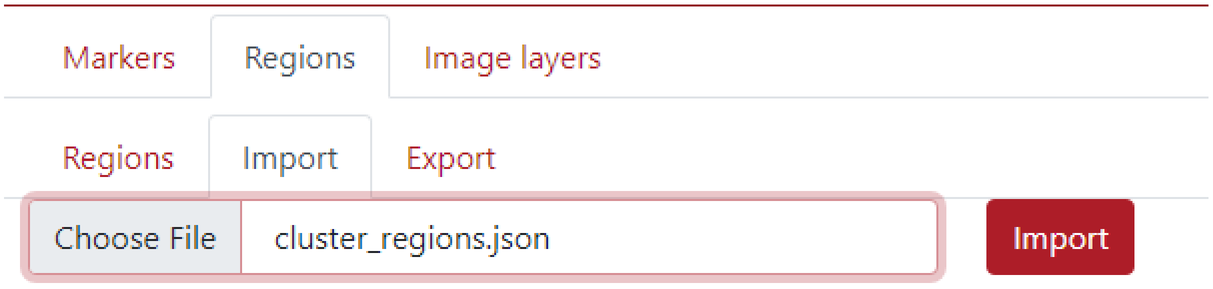


Fill	Name	Class	Color	Delete
☐	All	Hippocampus	🟩	🗑
☑	region1	Hippocampus	🟩	🗑

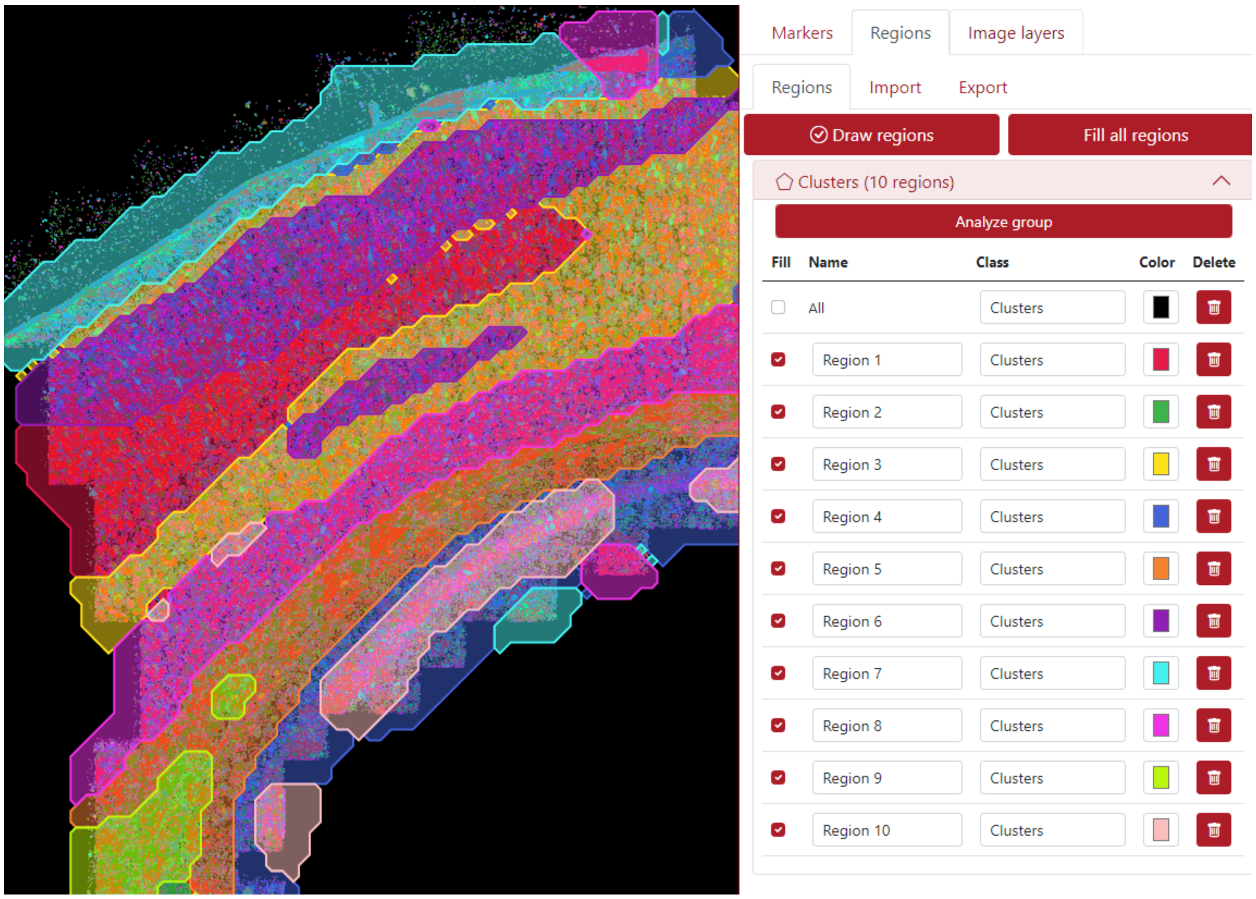
Key	Name	Count
Kcnip2	undefined	3222
Aldoc	undefined	1313
Lamp5	undefined	1033
Anln	undefined	639
Ctps	undefined	592
Mfge8	undefined	578
Gad2	undefined	498
Hexb	undefined	458
Cnr1	undefined	407
Gfap	undefined	405
Tbr1	undefined	397
Cpne5	undefined	256
Pthlh	undefined	191

2.3.4 Import Regions

Regions can be imported from .json file, which could be achieved from an external software or also from TissUUmaps' plugin *Points2Regions*. The user just click on the tab *Import* -> *Choose File* and press the button *Import*.



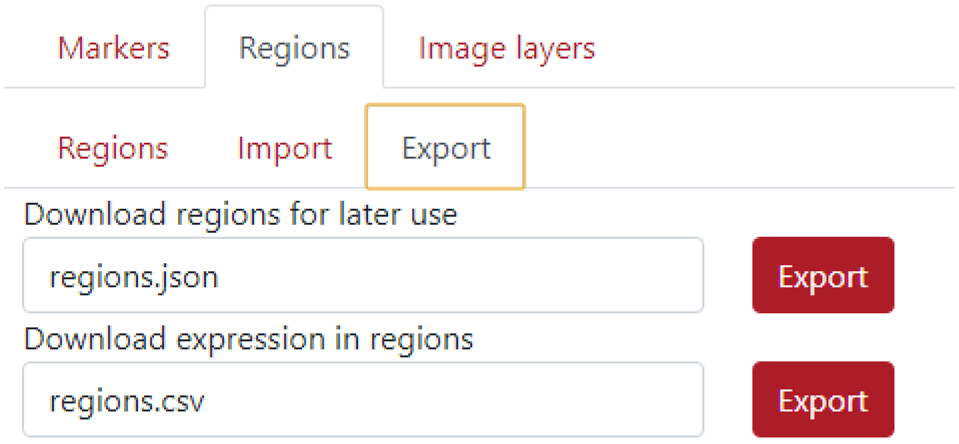
After that, the displayed regions appear in the left panel and the list of regions in the right panel as you can see in the example below. In this case, there are 10 different regions, called clusters. The user can change the color, the name, and the class of the regions if necessary. The user can as well draw some extra regions. These regions can be analyzed to observe the marker expression.



Fill	Name	Class	Color	Delete
☐	All	Clusters	⬛	🗑
✓	Region 1	Clusters	🟥	🗑
✓	Region 2	Clusters	🟩	🗑
✓	Region 3	Clusters	🟨	🗑
✓	Region 4	Clusters	🟦	🗑
✓	Region 5	Clusters	🟧	🗑
✓	Region 6	Clusters	🟪	🗑
✓	Region 7	Clusters	🟦	🗑
✓	Region 8	Clusters	🟪	🗑
✓	Region 9	Clusters	🟩	🗑
✓	Region 10	Clusters	🟥	🗑

2.3.5 Export Regions

The regions can be exported by clicking the tab *Export*, there the user can export two types of files. The first one is the .json file and the name can be selected. The second file is the marker expression in the regions which can be exported as .csv file (this is exported only if the regions were analyzed).



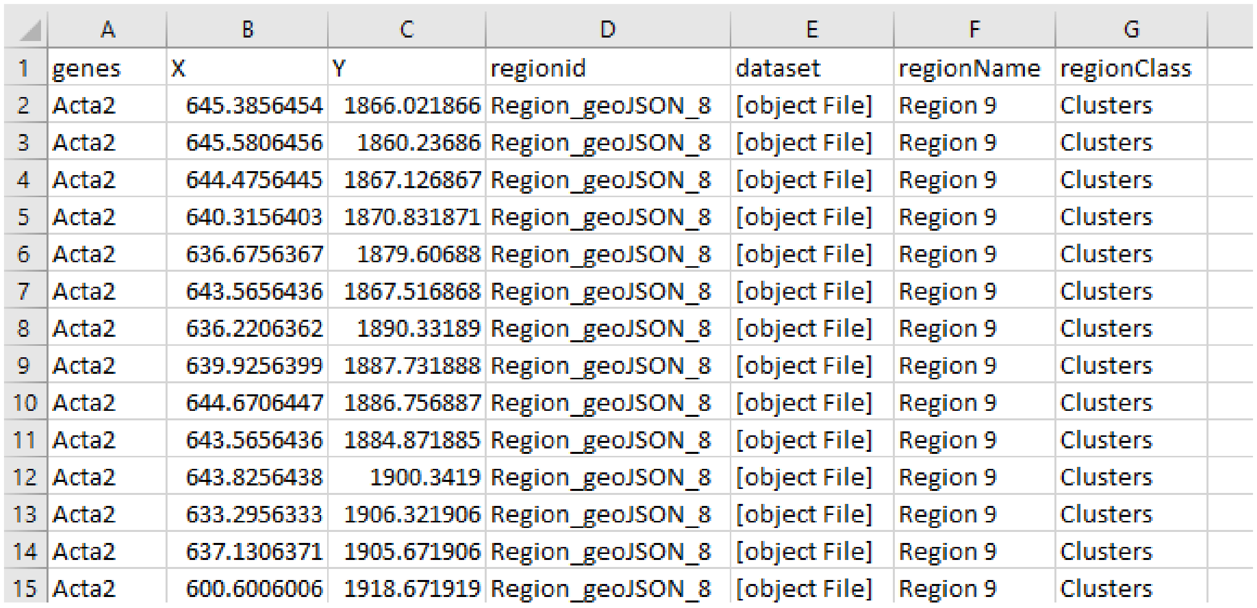
Download regions for later use

regions.json — Export

Download expression in regions

regions.csv — Export

In the figure below can be seen an example of the exported .cvs file.

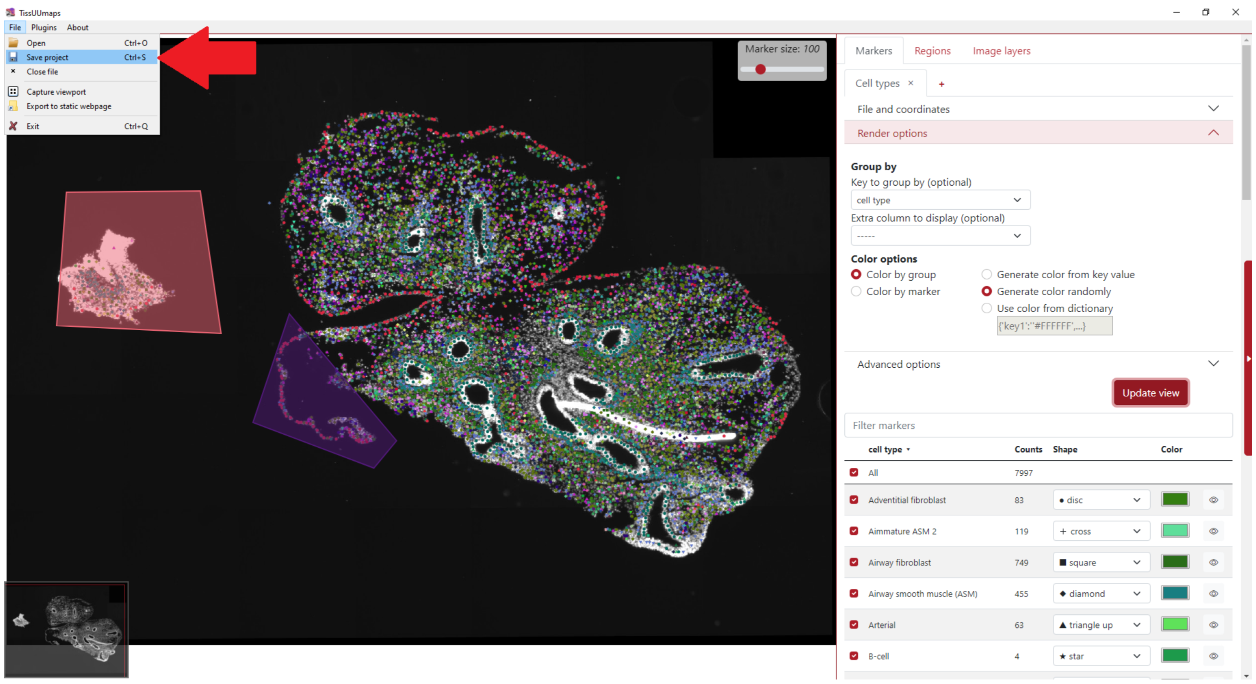


	A	B	C	D	E	F	G
1	genes	X	Y	regionid	dataset	regionName	regionClass
2	Acta2	645.3856454	1866.021866	Region_geoJSON_8	[object File]	Region 9	Clusters
3	Acta2	645.5806456	1860.23686	Region_geoJSON_8	[object File]	Region 9	Clusters
4	Acta2	644.4756445	1867.126867	Region_geoJSON_8	[object File]	Region 9	Clusters
5	Acta2	640.3156403	1870.831871	Region_geoJSON_8	[object File]	Region 9	Clusters
6	Acta2	636.6756367	1879.60688	Region_geoJSON_8	[object File]	Region 9	Clusters
7	Acta2	643.5656436	1867.516868	Region_geoJSON_8	[object File]	Region 9	Clusters
8	Acta2	636.2206362	1890.33189	Region_geoJSON_8	[object File]	Region 9	Clusters
9	Acta2	639.9256399	1887.731888	Region_geoJSON_8	[object File]	Region 9	Clusters
10	Acta2	644.6706447	1886.756887	Region_geoJSON_8	[object File]	Region 9	Clusters
11	Acta2	643.5656436	1884.871885	Region_geoJSON_8	[object File]	Region 9	Clusters
12	Acta2	643.8256438	1900.3419	Region_geoJSON_8	[object File]	Region 9	Clusters
13	Acta2	633.2956333	1906.321906	Region_geoJSON_8	[object File]	Region 9	Clusters
14	Acta2	637.1306371	1905.671906	Region_geoJSON_8	[object File]	Region 9	Clusters
15	Acta2	600.6006006	1918.671919	Region_geoJSON_8	[object File]	Region 9	Clusters

2.4 Projects

2.4.1 Saving projects

When the user has finished the visualization adjustments, region drawings, etc., the project is ready to be saved in order to continue working on it later or just basically to save it as it is for further consistency. The user needs to press *File* in the menu and then *Save project* or Ctrl + S.



Group by

Key to group by (optional): cell type

Extra column to display (optional): -----

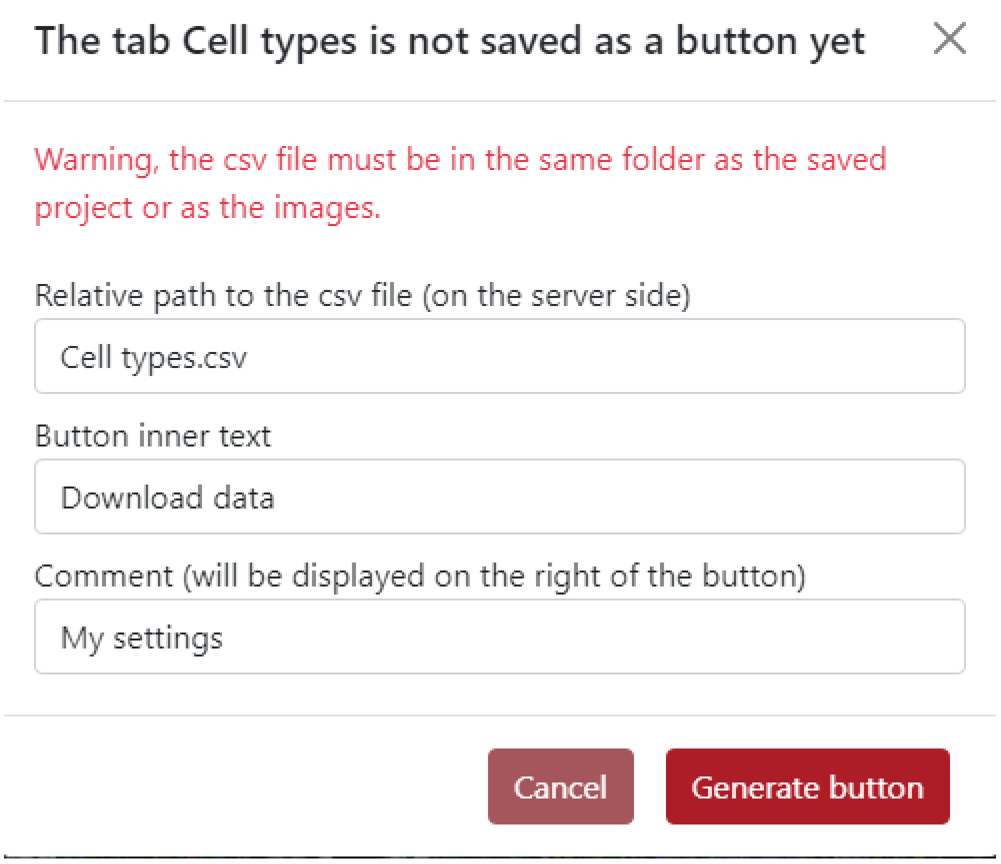
Color options

[✓ Color by group] Color by marker

Generate color from key value [✓ Generate color randomly] Use color from dictionary {'key1':'#FFFFFF',...}

cell type	Counts	Shape	Color
All	7997		
Adventitial fibroblast	83	● disc	🟩
Aimmature ASM 2	119	+ cross	🟩
Airway fibroblast	749	■ square	🟩
Airway smooth muscle (ASM)	455	◆ diamond	🟦
Arterial	63	▲ triangle up	🟩
B-cell	4	★ star	🟩

In order to save the project together with the .csv file, it is necessary to generate a button first. The warning window below appears and the user needs to generate the button. The path to the .csv file needs to be relative to the path of the image. In this example, the image layer and the .csv file are in the exact same directory.



The tab Cell types is not saved as a button yet

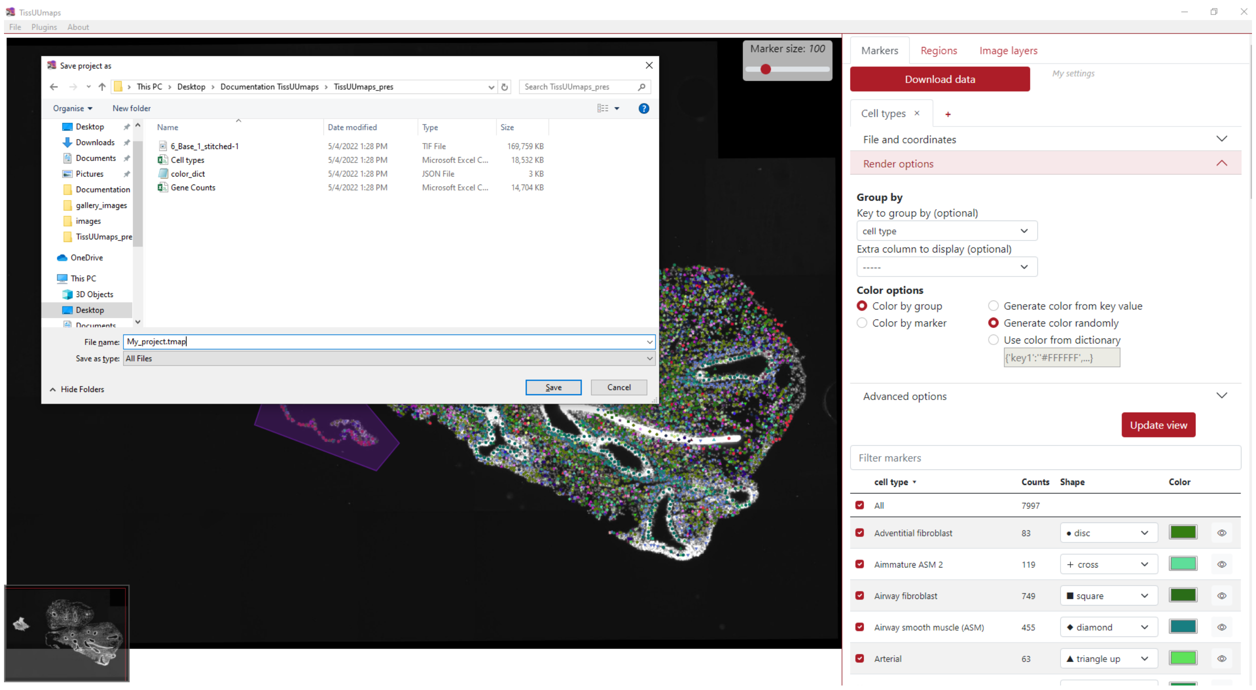
Warning, the csv file must be in the same folder as the saved project or as the images.

Relative path to the csv file (on the server side): Cell types.csv

Button inner text: Download data

Comment (will be displayed on the right of the button): My settings

Then the user selects a suitable directory to save the project and writes the project file name, i.e. My_project.tmap, and the project is saved.



Name	Date modified	Type	Size
6_Base_1_stitched-1	5/4/2022 1:28 PM	TIF File	169,759 KB
Cell types	5/4/2022 1:28 PM	Microsoft Excel C...	18,532 KB
color_dict	5/4/2022 1:28 PM	JSON File	3 KB
Gene Counts	5/4/2022 1:28 PM	Microsoft Excel C...	14,704 KB

File name: My_project.tmap

Save as type: All Files

My settings

Group by

Key to group by (optional): cell type

Extra column to display (optional): -----

Color options

[✓ Color by group] Color by marker

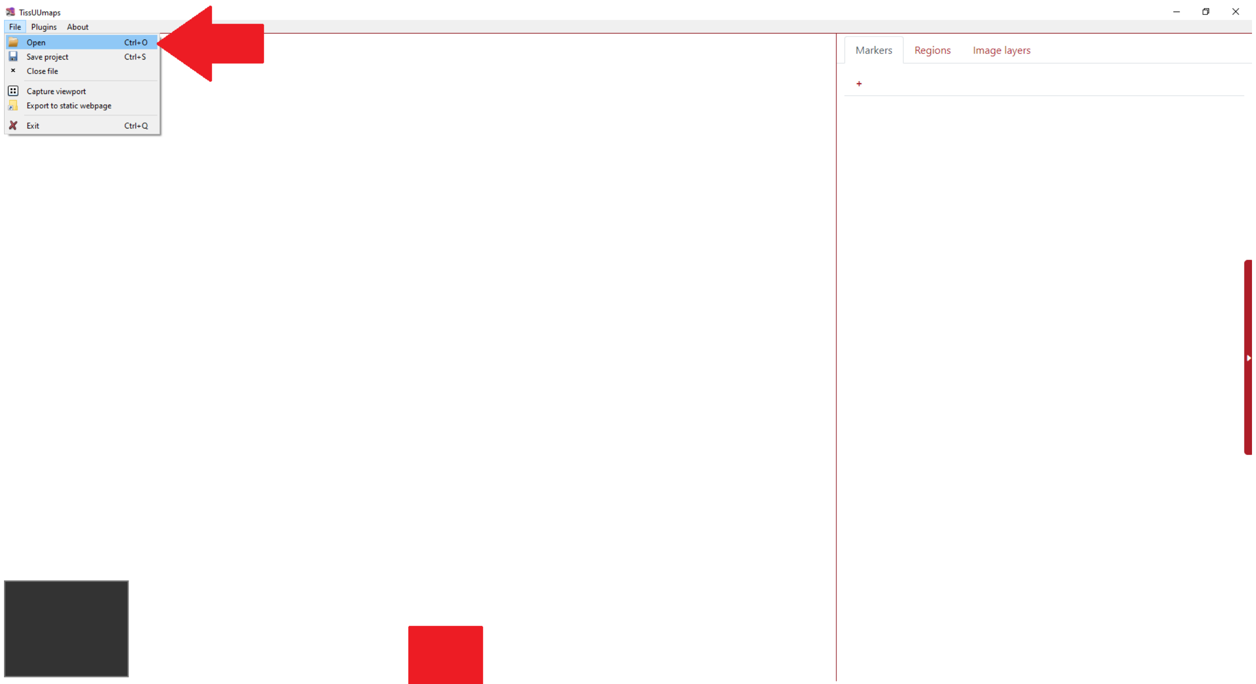
Generate color from key value [✓ Generate color randomly] Use color from dictionary

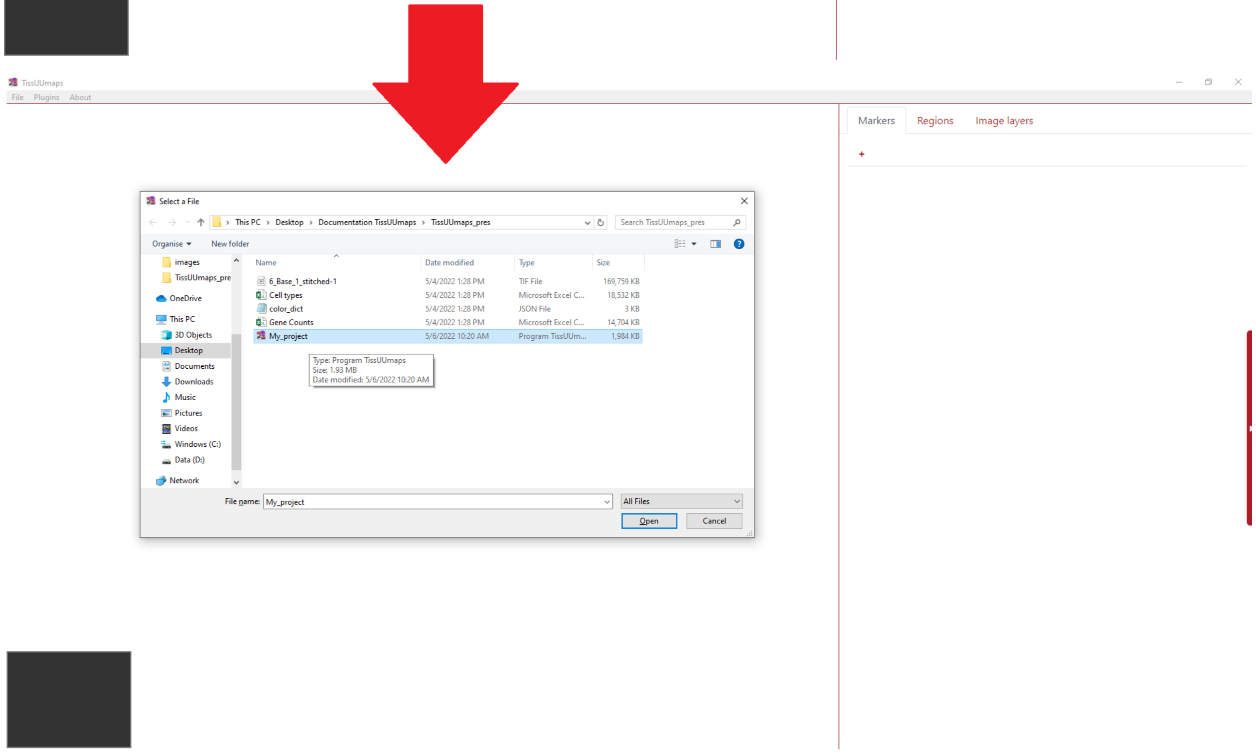
{'key1':'#FFFFFF',...}

	cell type	Counts	Shape	Color
✓	All	7997		
✓	Adventitial fibroblast	83	● disc	🟩
✓	Aimmature ASM 2	119	+ cross	🟩
✓	Airway fibroblast	749	■ square	🟩
✓	Airway smooth muscle (ASM)	455	◆ diamond	🟦
✓	Arterial	63	▲ triangle up	🟩

2.4.2 Loading projects

The .tmap project can be loaded by two approaches. The first one is opening the TissUUmaps program, click *File* in the menu and then *Open* or Crtl + O. Then the user navigates in the directory and selects the .tmap file. By default, the directory navigates in the recent .tmap project.



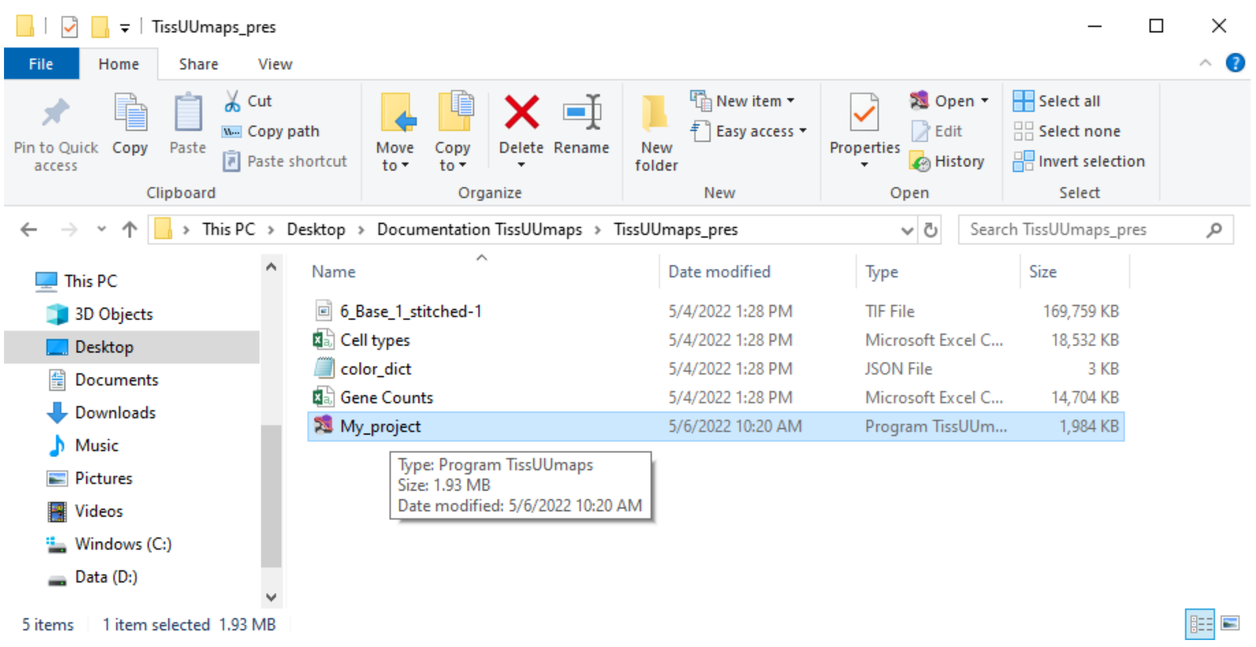


Name	Date modified	Type	Size
6_Base_1_stitched-1	5/4/2022 1:28 PM	TIF File	169,759 KB
Cell types	5/4/2022 1:28 PM	Microsoft Excel C...	18,532 KB
color_dict	5/4/2022 1:28 PM	JSON File	3 KB
Gene Counts	5/4/2022 1:28 PM	Microsoft Excel C...	14,704 KB
My_project	5/6/2022 10:20 AM	Program TissUUm...	1,984 KB

Type: Program TissUUmaps
Size: 1.93 MB
Date modified: 5/6/2022 10:20 AM

File name: My_project
All Files

The second option is directly double click on the .tmap file in file explorer in your computer.

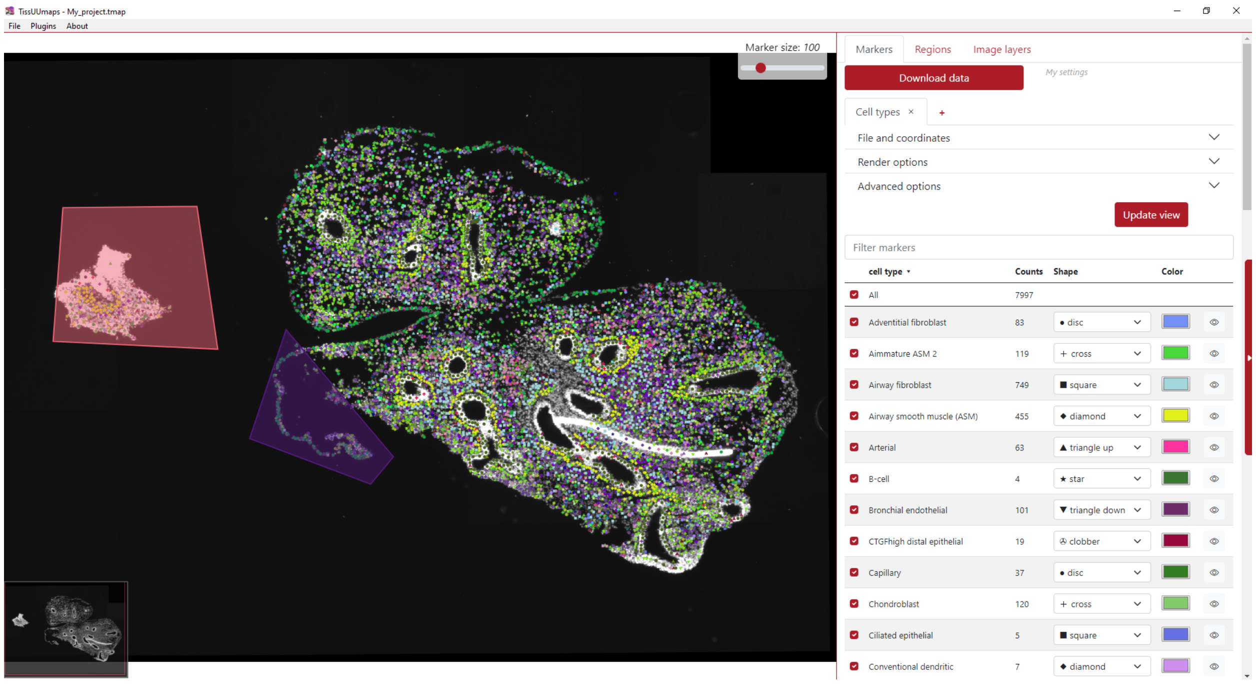


Name	Date modified	Type	Size
6_Base_1_stitched-1	5/4/2022 1:28 PM	TIF File	169,759 KB
Cell types	5/4/2022 1:28 PM	Microsoft Excel C...	18,532 KB
color_dict	5/4/2022 1:28 PM	JSON File	3 KB
Gene Counts	5/4/2022 1:28 PM	Microsoft Excel C...	14,704 KB
My_project	5/6/2022 10:20 AM	Program TissUUm...	1,984 KB

Type: Program TissUUmaps
Size: 1.93 MB
Date modified: 5/6/2022 10:20 AM

5 items   1 item selected 1.93 MB

After clicking the button *Download data*, both these approaches will lead to loading the project as can be seen in the example below.



	cell type	Counts	Shape	Color
✓	All	7997		
✓	Adventitial fibroblast	83	● disc	🟦
✓	Aimmature ASM 2	119	+ cross	🟩
✓	Airway fibroblast	749	■ square	🟦
✓	Airway smooth muscle (ASM)	455	◆ diamond	🟨
✓	Arterial	63	▲ triangle up	🟥
✓	B-cell	4	★ star	🟩
✓	Bronchial endothelial	101	▼ triangle down	🟪
✓	CTGFhigh distal epithelial	19	☮ clobber	🟥
✓	Capillary	37	● disc	🟩
✓	Chondroblast	120	+ cross	🟩
✓	Ciliated epithelial	5	■ square	🟦
✓	Conventional dendritic	7	◆ diamond	🟪

For more information on the tmap file format and specifications, see *The TMAP file format*.

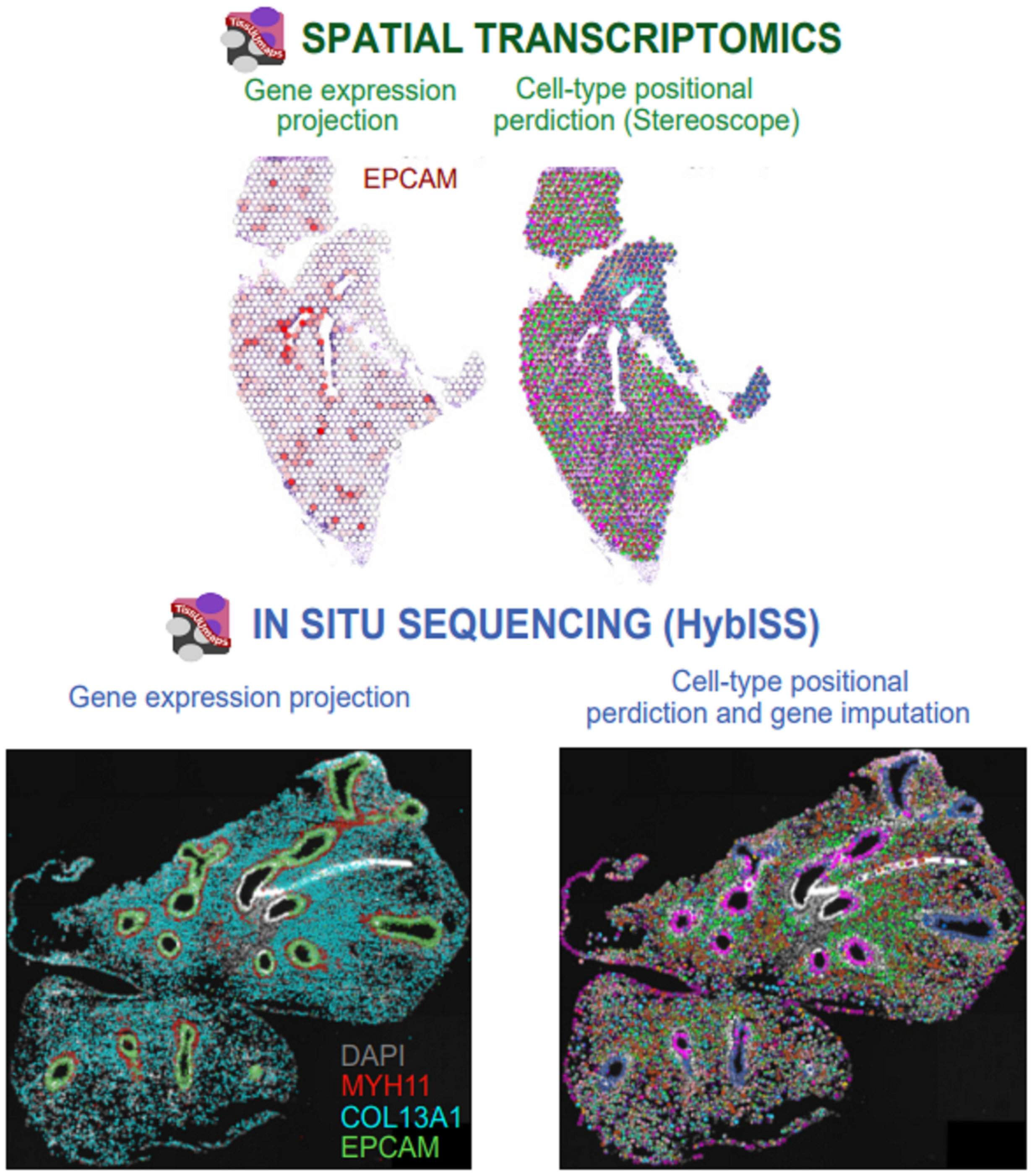
2.4.3 Editing .tmap file manually

Work in progress

2.4.4 Existing projects

Human Developmental Lung Cell Atlas (pcw 5- pcw 14)

The human lung is a highly complex tubular organ, whose main function is the gas exchange between blood and breathed air. In contains a large number of specialized cell-types of epithelial, endothelial, neuronal, stromal and immune cells that are necessary for normal organ function and structural integrity. To understand how this cell heterogeneity develops to create a healthy mature lung, we focused on the 1st trimester of gestation and applied state of art technologies to capture the gene expression profiles of all the cells in the developing organ, in time and space.


SPATIAL TRANSCRIPTOMICS
Gene expression projection
Cell-type positional perdiction (Stereoscope)
EPCAM
IN SITU SEQUENCING (HybISS)
Gene expression projection
Cell-type positional perdiction and gene imputation
DAPI
MYH11
COL13A1
EPCAM

TissUUmaps interactive viewer: Single-cell RNA-sequencing UMAP representation of single-cell clusters and sub-clusters, gene expression and metadata.

In situ sequencing data (ISS) - TissUUmaps interactive viewer: pcw 5 pcw 6 pcw 13In situ sequencing data. Spot location + identity, per bin pie chart view of cell type probabilities and imputed genes.

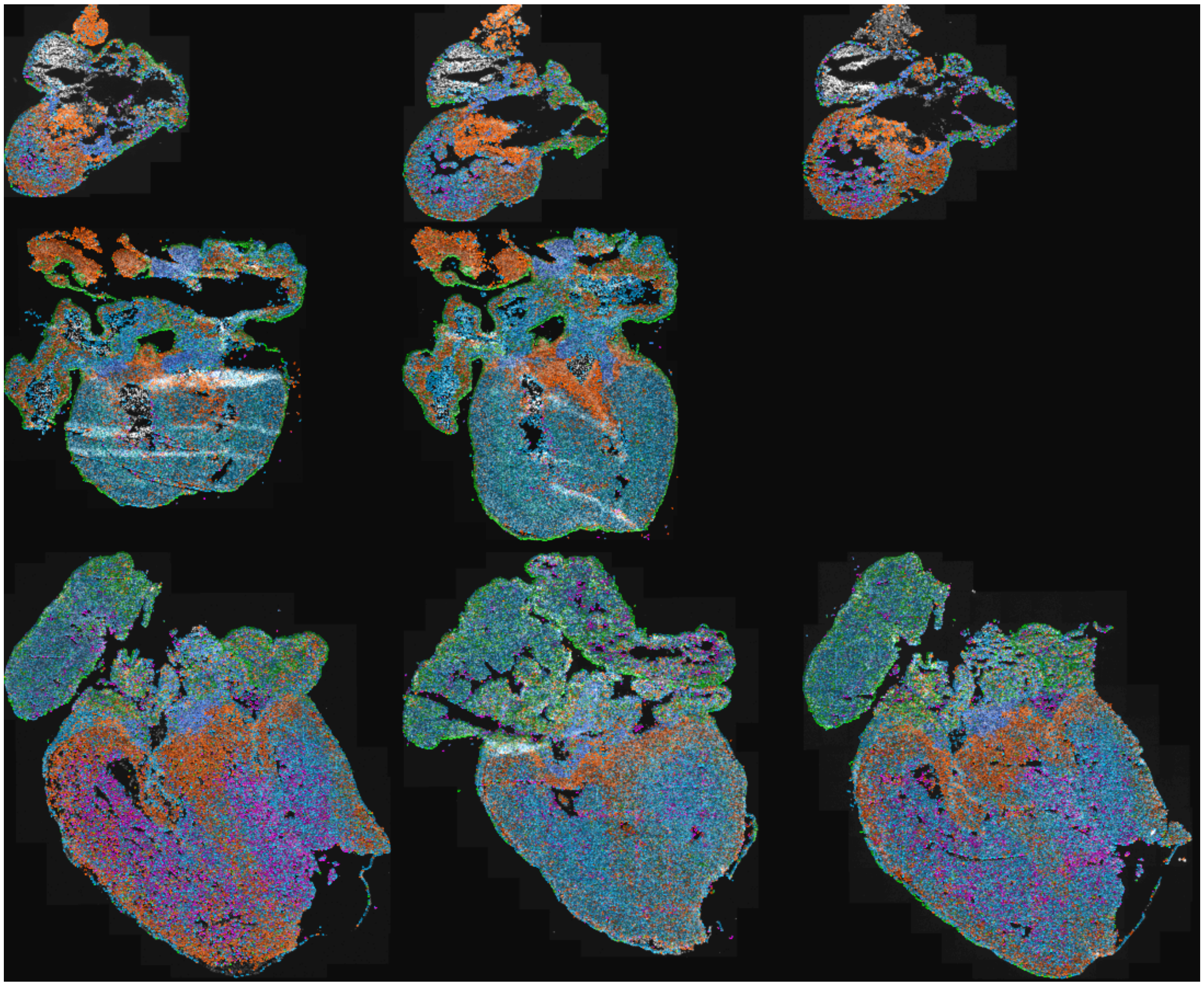
SCRINSHOT data - TissUUmaps interactive viewer: pcw 6 pcw 8 pcw 11 pcw 14SCRINSHOT data. Spot location + identity.

Spatial Transcriptomics data - TissUUmaps interactive viewer: pcw 6 pcw 8 pcw 10 pcw 11Per gene or pie chart view of gene expression.

More information is available in the original publication: A. Sountoulidis, S.M. Salas, E. Braun, C. Avenel, J. Bergenstråhle, M. Vicari, P. Czarnewski, J. Theelke, A. Liontos, X. Abalo, Ž. Andrusivová, M. Asp, X. Li, L. Hu, S. Sariyar, A.M. Casals, B. Ayoglu, A. Firsova, J. Michaëlsson, E. Lundberg, C. Wählby, E. Sundström, S. Linnarsson, J. Lundeberg, M. Nilsson, C. Samakovlis. Developmental origins of cell heterogeneity in the human lung. BioRxiv doi: https://doi.org/10.1101/2022.01.11.475631

Modelling of cell-type signatures in the developmental human heart

With the emergence of high throughput single cell techniques, the understanding of cellular diversity in biologically complex processes has rapidly increased. The next step towards comprehension of e.g. key organs in the mammal development is to obtain spatiotemporal atlases of the cellular diversity. However, targeted cell typing approaches relying on existing single cell data achieve incomplete and biased maps that could mask the molecular and cellular heterogeneity present in a tissue slide. Here we applied spage2vec, a de novo approach to spatially resolve and characterize cellular diversity during human heart development. Data from the original in situ sequencing experiment as well as identified cell types can be viewed in TissUUmaps.

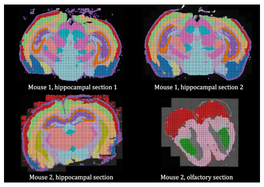


TissUUmaps interactive viewer: Human heart

More information is available in the original publication: RS. Marco Salas, X. Yuan, C. Sylven, M. Nilsson, C. Wählby and G.Partel. De novo spatiotemporal modelling of cell-type signatures identifies novel cell populations in the developmental human heart. BioRxiv doi: https://doi.org/10.1101/2021.07.10.451822

Automated identification of the mouse brain's spatial compartments from in situ sequencing data

Neuroanatomical compartments of the mouse brain are identified and outlined mainly based on manual annotations of samples using features related to tissue and cellular morphology, taking advantage of publicly available reference atlases. However, this task is challenging since sliced tissue sections are rarely perfectly parallel or angled with respect to sections in the reference atlas and organs from different individuals may vary in size and shape and requires manual annotation. Here, we show how in situ sequencing data combined with dimensionality reduction and unsupervised clustering can be used to identify spatial compartments that correspond to known anatomical compartments of the brain. Here we show results on four different sections of mouse brains.


Mouse 1, hippocampal section 1
Mouse 1, hippocampal section 2
Mouse 2, hippocampal section
Mouse 2, olfactory section

TissUUmaps interactive viewer: Mouse brain

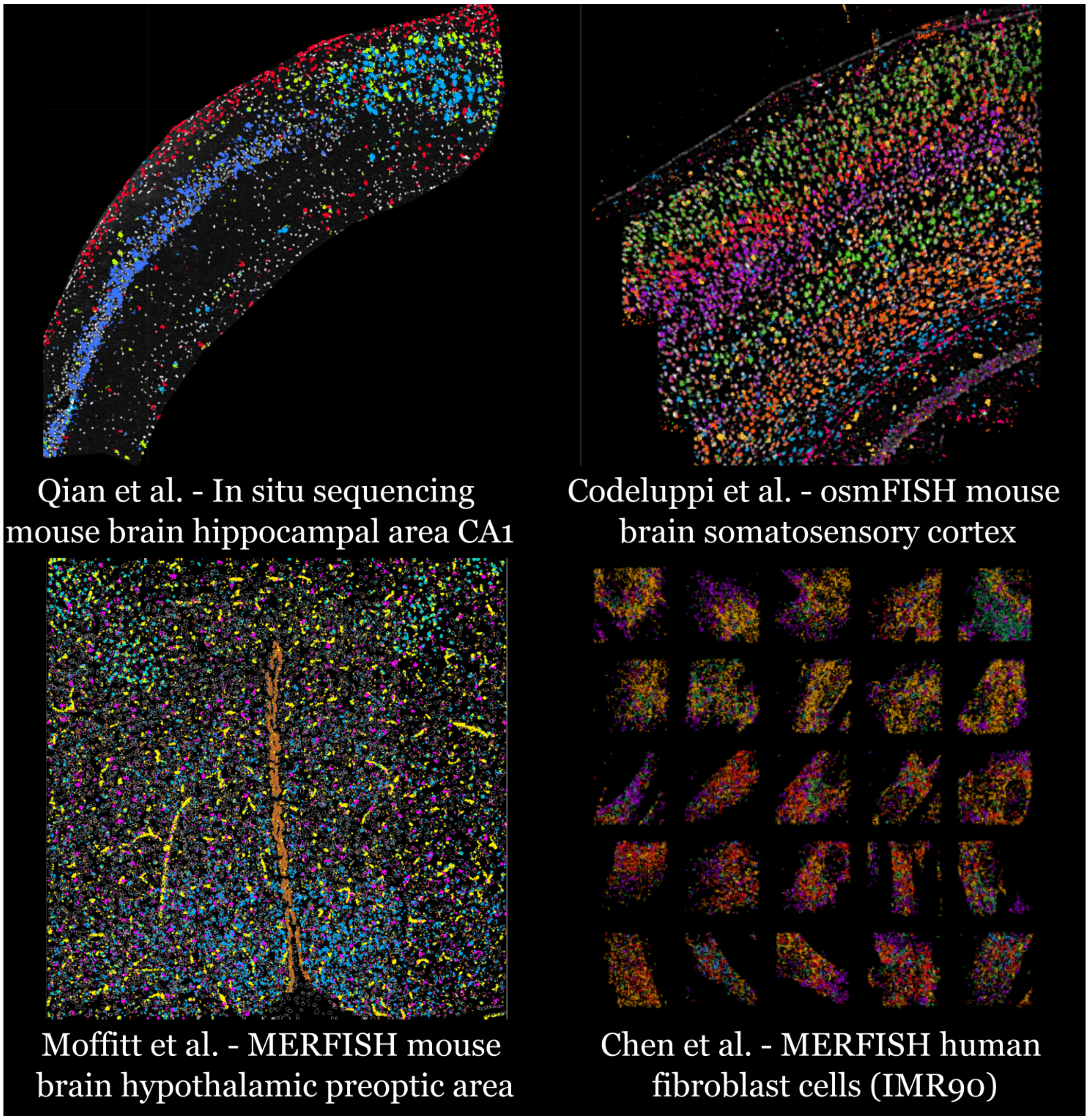
More information is available in this publication: G. Partel, M.M. Hilscher, G. Milli, L. Solorzano, A.H. Klemm, M. Nilsson, and C. Wählby. Automated identification of the mouse brain's spatial compartments from in situ sequencing data. BMC Biology, https://doi.org/10.1186/s12915-020-00874-5, Oct 2020.

The original raw ISS data was published in Qian, X., Harris, K. D., Hauling, T., Nicoloutsopoulos, D., Muñoz-Manchado, A. B., Skene, N., … & Nilsson, M. (2020). Probabilistic cell typing enables fine mapping of closely related cell types in situ. Nature methods, 17(1), 101-106.

Data and code availability: All software was developed in Python 3 using open source libraries, and data processing of pipeline workflows was carried out using Anduril2 analysis framework. The processing pipelines, data, and the software version used to generate the analysis results and figures presented in this paper are available at https://doi.org/10.5281/zenodo.3928219 or from our github repository https://github.com/wahlby-lab/graph-iss.

Spage2vec: Unsupervised representation of localized spatial gene expression signatures

Spage2vec is an unsupervised segmentation free approach for decrypting the spatial transcriptomic heterogeneity of complex tissues at subcellular resolution. Spage2vec represents the spatial transcriptomic landscape of tissue samples as a graph and leverage powerful machine learning graph representation technique to create a lower dimensional representation of local spatial gene expression. Here we visualize spage2vec localized gene expression signatures of different spatial transcriptomic datasets. We thank Mats Nilsson, Sten Linnarsson and Xiaowei Zhuang for making their datasets publicly available.


Qian et al. - In situ sequencing mouse brain hippocampal area CA1
Codeluppi et al. - osmFISH mouse brain somatosensory cortex
Moffitt et al. - MERFISH mouse brain hypothalamic preoptic area
Chen et al. - MERFISH human fibroblast cells (IMR90)

TissUUmaps interactive viewer 1: In situ sequencing mouse brain hippocampal area CA1 Qian, X., Harris, K. D., Hauling, T., Nicoloutsopoulos, D., Muñoz-Manchado, A. B., Skene, N., … & Nilsson, M. (2020). Probabilistic cell typing enables fine mapping of closely related cell types in situ. Nature methods, 17(1), 101-106.

TissUUmaps interactive viewer 2: osmFISH mouse brain somatosensory cortex Codeluppi, S., Borm, L. E., Zeisel, A., La Manno, G., van Lunteren, J. A., Svensson, C. I., & Linnarsson, S. (2018). Spatial organization of the somatosensory cortex revealed by osmFISH. Nature methods, 15(11), 932-935.

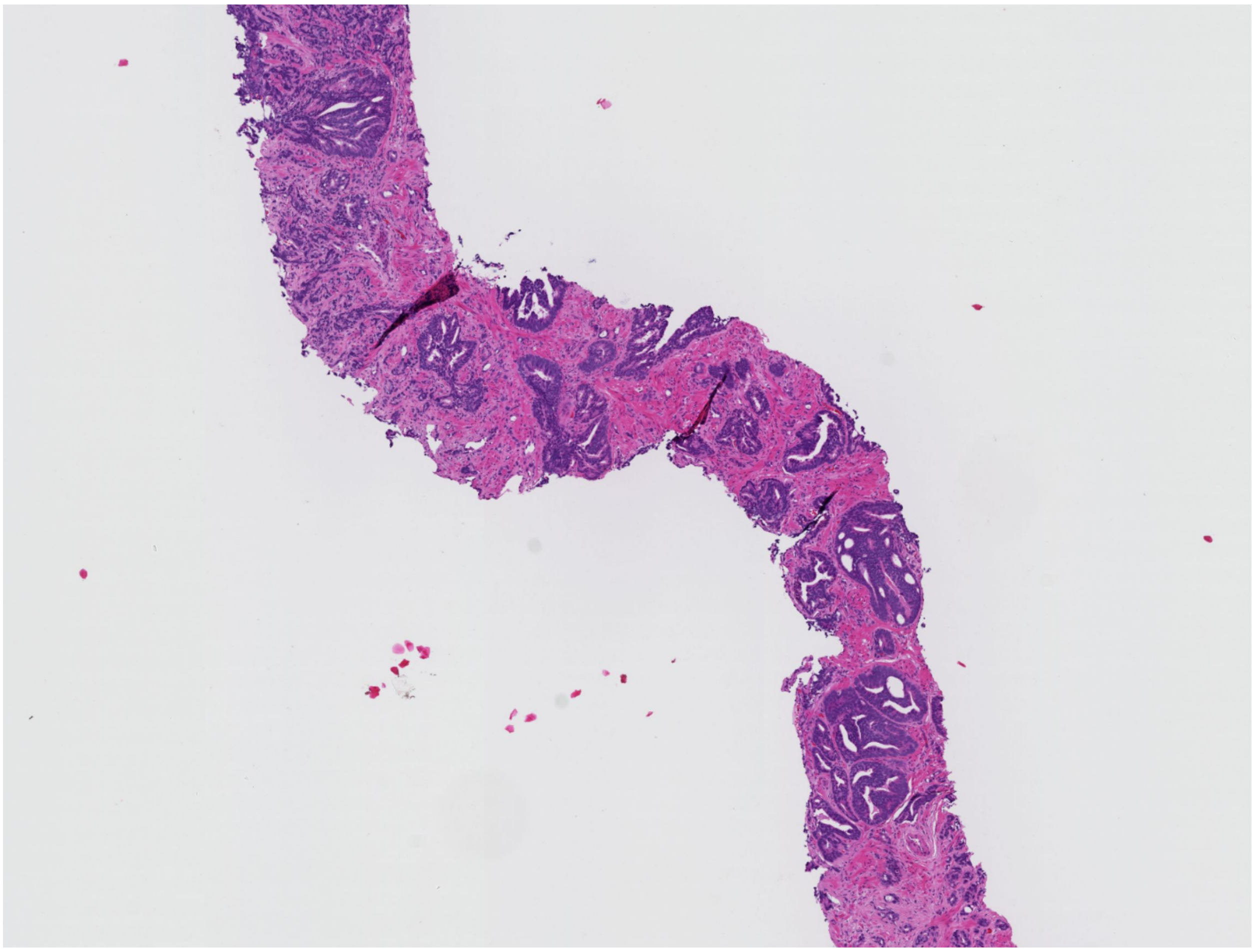
TissUUmaps interactive viewer 3: MERFISH mouse brain hypothalamic preoptic area Moffitt, J. R., Bambah-Mukku, D., Eichhorn, S. W., Vaughn, E., Shekhar, K., Perez, J. D., … & Zhuang, X. (2018). Molecular, spatial, and functional single-cell profiling of the hypothalamic preoptic region. Science, 362(6416), eaau5324.

TissUUmaps interactive viewer 4: MERFISH human fibroblast cells (IMR90) Chen, K. H., Boettiger, A. N., Moffitt, J. R., Wang, S., & Zhuang, X. (2015). Spatially resolved, highly multiplexed RNA profiling in single cells. Science, 348(6233), aaa6090.

Data and code availability: Spatial gene expression data are available in Zenodo database at https://doi.org/10.5281/zenodo.3897401. Source code for reproducing analysis results and figures is available in Zenodo database at http://www.doi.org/10.5281/zenodo.4030404.

Artificial intelligence for diagnosis and grading of prostate cancer in biopsies: a population-based

An increasing volume of prostate biopsies and a worldwide shortage of urological pathologists puts a strain on pathology departments. Additionally, the high intra-observer and inter-observer variability in grading can result in overtreatment and undertreatment of prostate cancer. To alleviate these problems, we aimed to develop an artificial intelligence (AI) system with clinically acceptable accuracy for prostate cancer detection, localisation, and Gleason grading. Here we show examples of full-resolution digitized biopsies and corresponding AI-based grading.



An overview of all sample **datasets** can be found here: Prostate cancer in biopsies

More information is available in this publication: P. Ström, K. Kartasalo, H. Olsson, L. Solorzano et al. Artificial intelligence for diagnosis and grading of prostate cancer in biopsies: a population-based, diagnostic study. The Lancet Oncology, Volume 21, Issue 2, 2020, Pages 222-232, ISSN 1470-2045, doi: 10.1016/S1470-2045(19)30738-7, url: https://www.sciencedirect.com/science/article/pii/S1470204519307387

2.5 Exporting screenshots

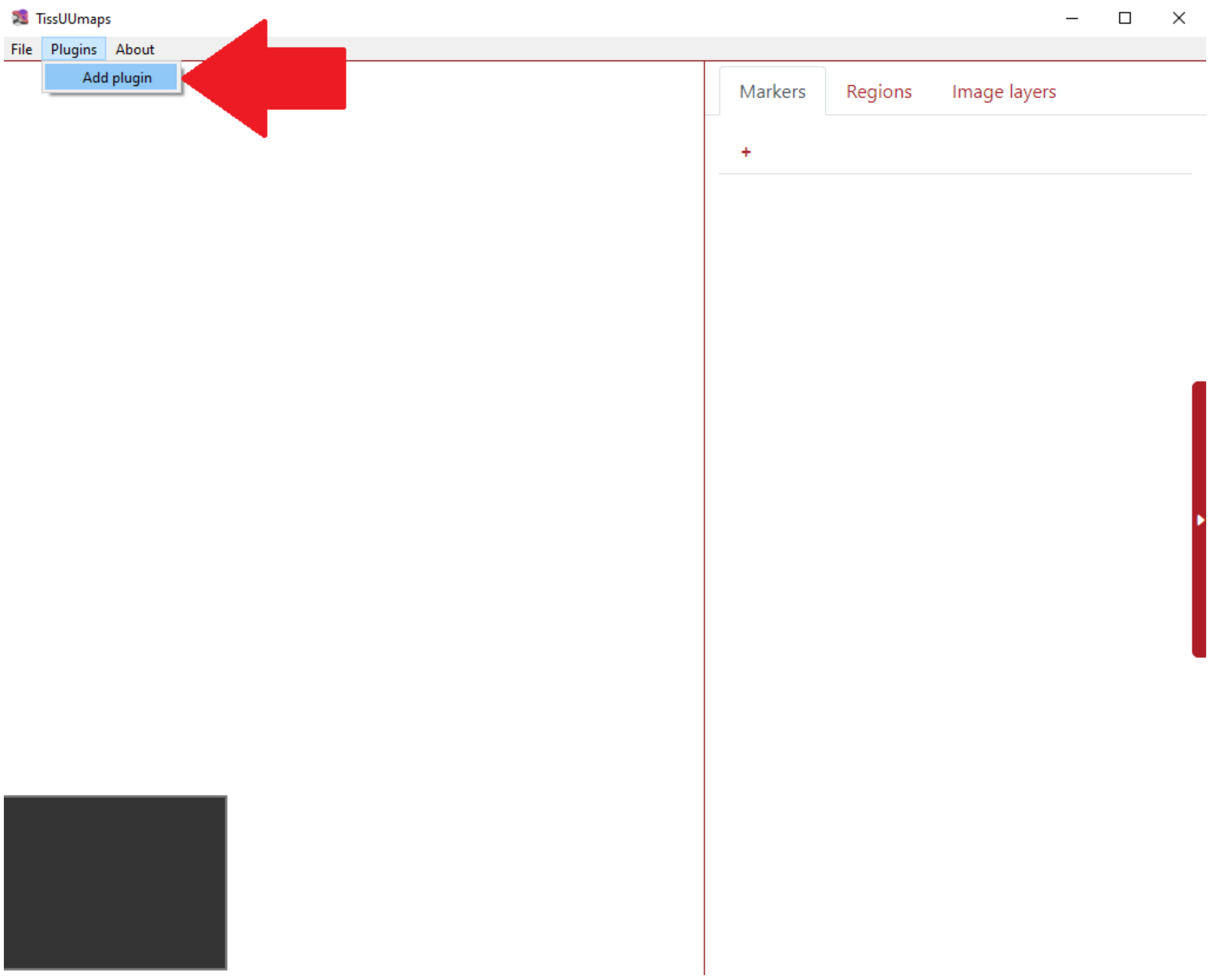
TissUUmaps allows high resolution capture of the image viewport. Go to `Menu > File > Capture viewport` and chose a zoom factor for export (1 = screen resolution).

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.

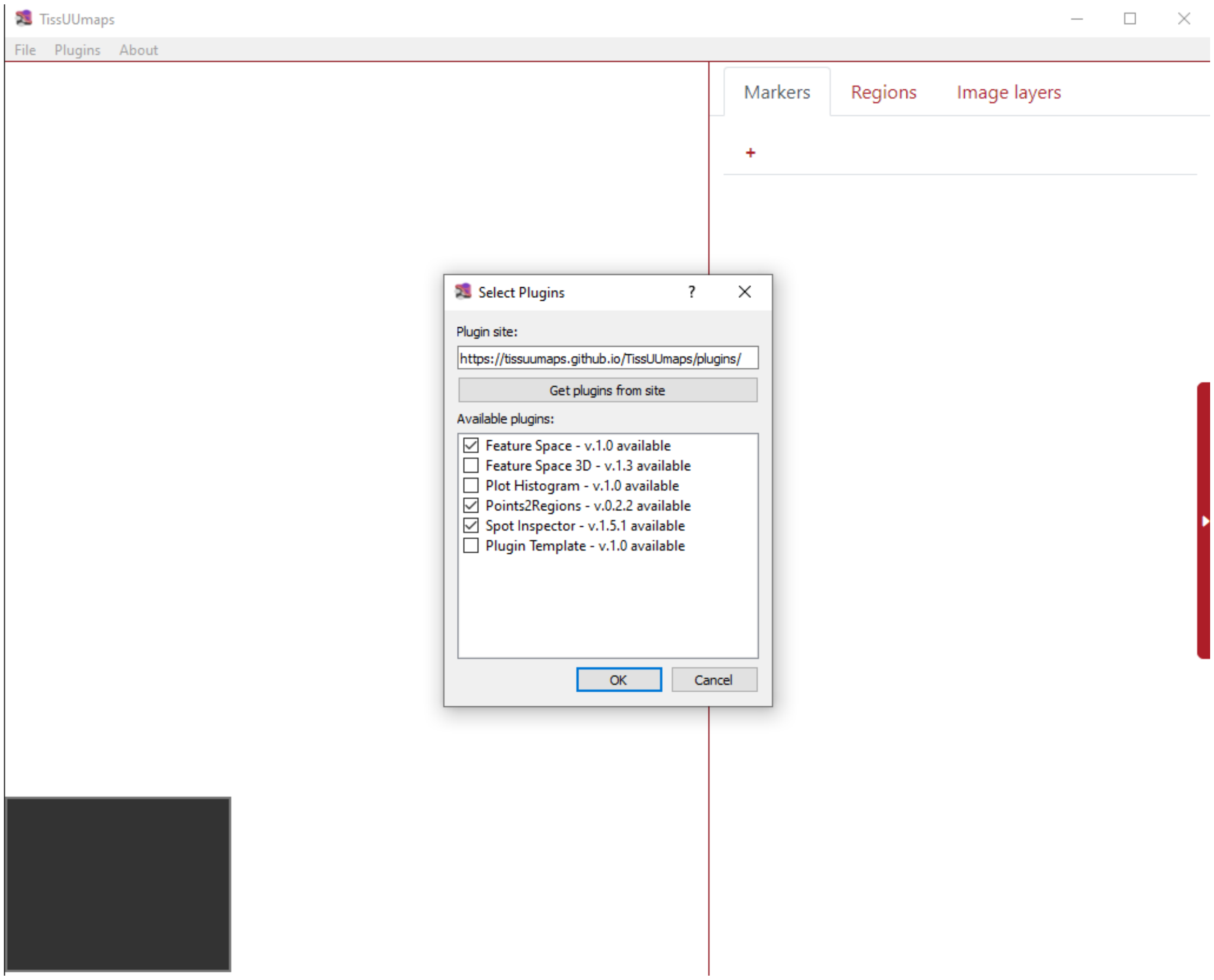
2.6 Plugins

2.6.1 Load plugins

In order to load plugins, first, they need to be installed. This can be done in the menu `Plugins > Add plugin` as can be seen in the example below.



Consequently, the user can check any number of plugins they desire and press *OK*.



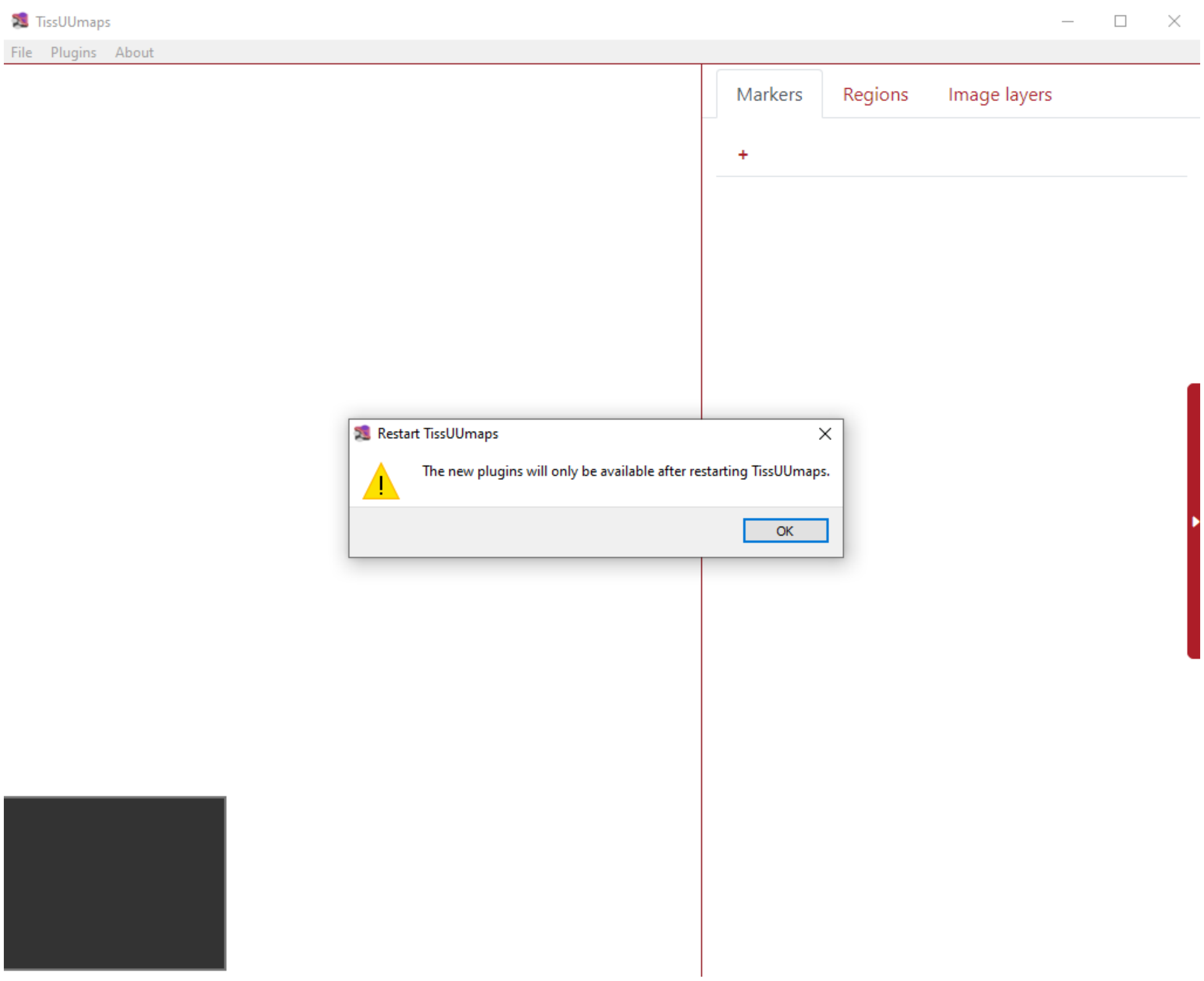
Plugin site:

https://tissuumaps.github.io/TissUUmaps/plugins/

Available plugins:

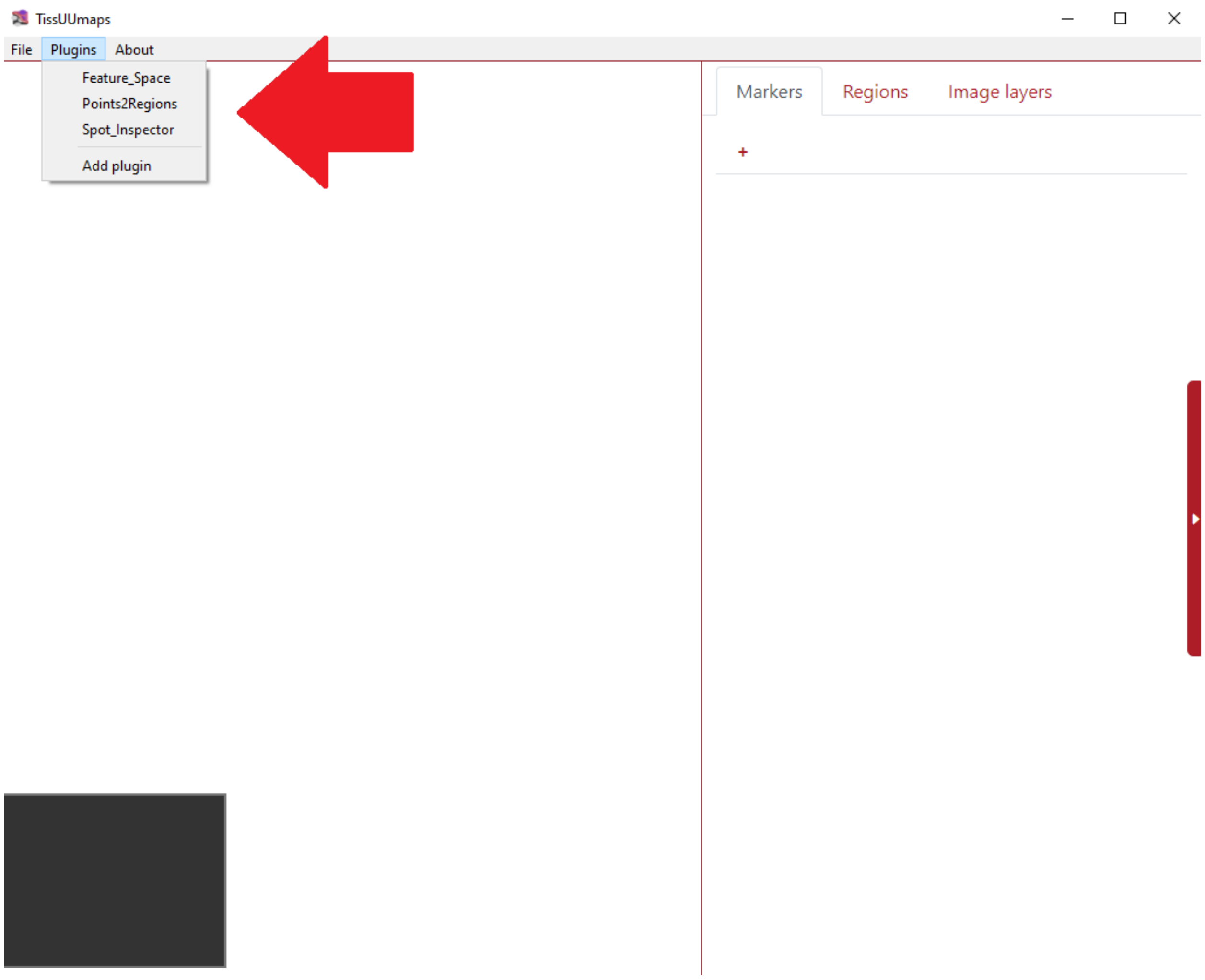
- [✓] Feature Space - v.1.0 available
- [ ] Feature Space 3D - v.1.3 available
- [ ] Plot Histogram - v.1.0 available
- [✓] Points2Regions - v.0.2.2 available
- [✓] Spot Inspector - v.1.5.1 available
- [ ] Plugin Template - v.1.0 available

Then the tool warns the user then the installed plugins will be available after restarting TissUUmaps.

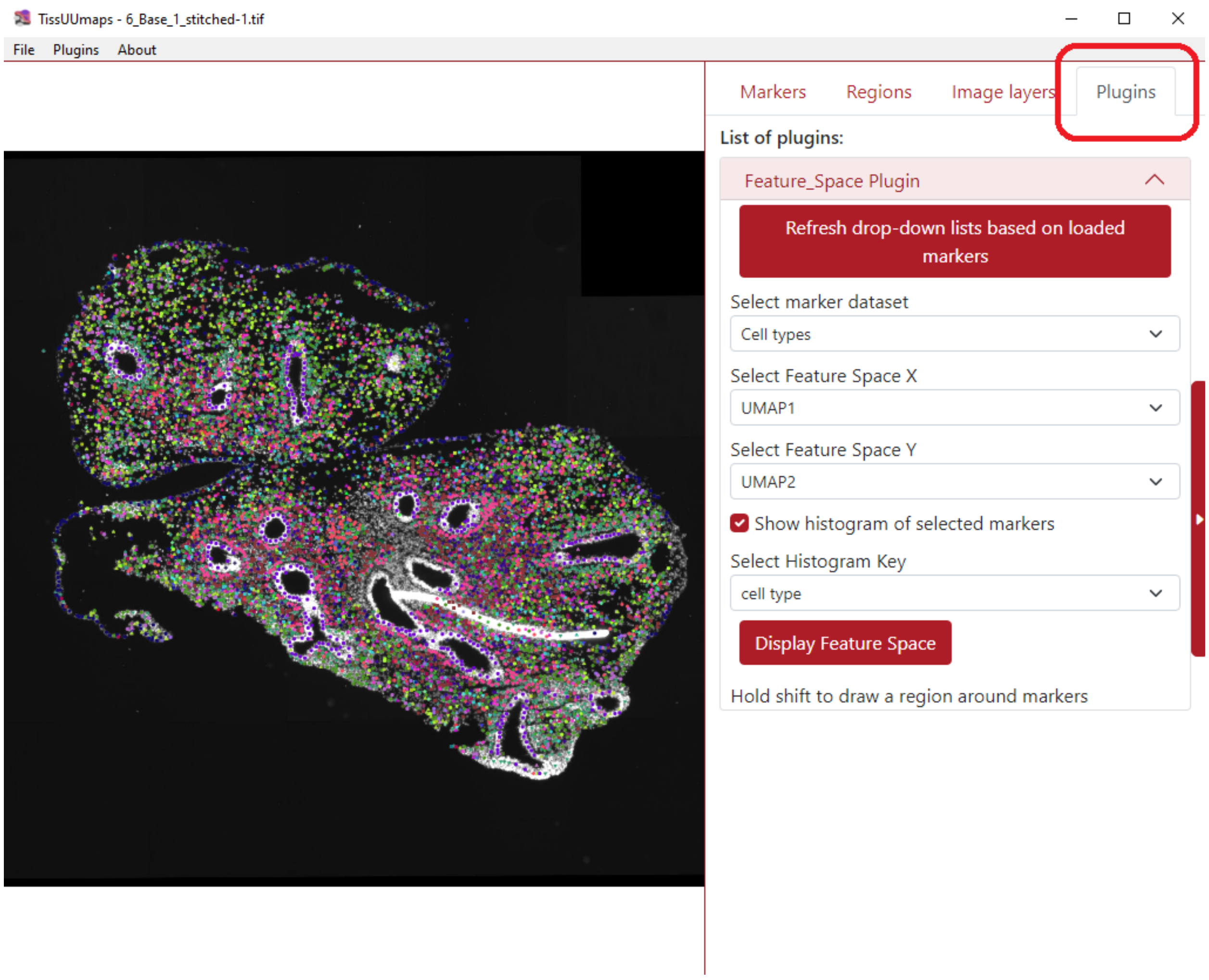


The new plugins will only be available after restarting TissUUmaps.

After restarting the TissUUmaps, all the installed plugins are listed in the menu `Plugins` as you can see in the figure below.



Once the user selects any of the installed plugins (in the example below I selected *Feature_Space*), a new tab *Plugins* appears in the upper right part of the screen with all the required boxes for filling.



List of plugins:

Feature_Space Plugin

Refresh drop-down lists based on loaded markers

Select marker dataset: Cell types

Select Feature Space X: UMAP1

Select Feature Space Y: UMAP2

☑ Show histogram of selected markers

Select Histogram Key: cell type

Display Feature Space

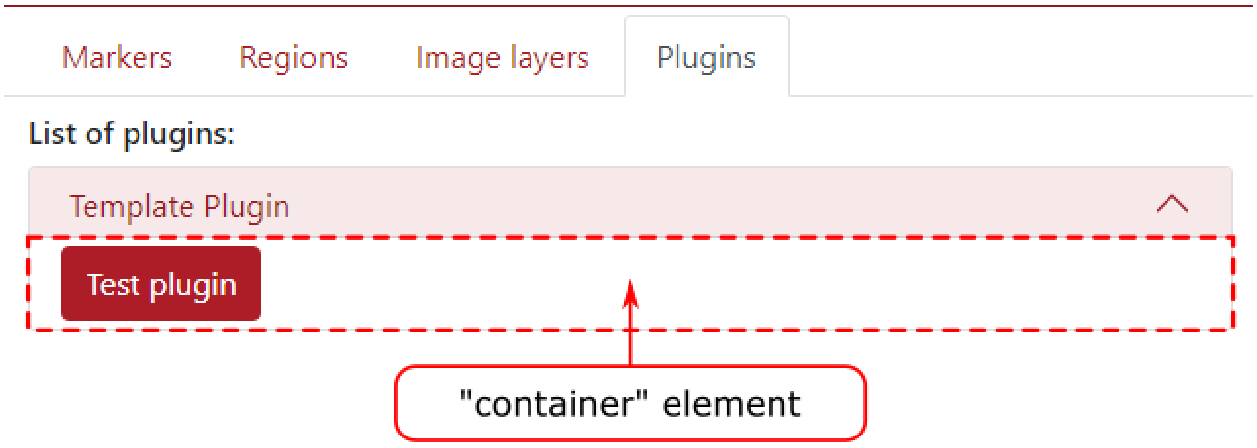
Hold shift to draw a region around markers

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.

2.7 Shortcuts

TissUUmaps contains following keyboard shorcuts:

`M` - to toggle the markers

`F` - to expand to fullscreen

`0` - to hide right menu

`R` - to toggle regions

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

3.2.1 Start a TissUUmaps docker instance

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.

3.2.2 Define TissUUmaps service in a Compose file

Compose is a tool for defining and running multi-container Docker applications. With Compose, you use a YAML file to configure your application's services. Then, with a single command, you create and start all the services from your configuration.

If you use compose, you don't need to start the docker container from the previous section.

1. Install docker-compose.
2. Create a file called `docker-compose.yml` in your project directory and paste the following:

```
version: "3.9"
services:
  backend:
    image: docker.io/cavenel/tissuumaps:latest
    volumes:
```

(continues on next page)

(continued from previous page)

```
      - type: bind
        source: /jail
        target: /mnt/data
    restart: on-failure
    ports:
      - 127.0.0.1:8050:80
```

This Compose file defines TissUUmaps backend service on port 8050. The web service uses an image that's built from docker.io hub. It then binds the container and the host machine to the exposed port, 8050.

3. Put your data in the source folder (here `/jail/shared/`). You can change the source path in the `docker-compose.yml` file.

4. From your project directory, start up your application by running `docker compose up`.

```
$ docker-compose up
Creating network "tmap_compose_default" with the default driver
Creating tmap_compose_backend_1 ... done
Attaching to tmap_compose_backend_1
backend_1  | [2023-03-27 11:23:57 +0000] [1] [INFO] Starting gunicorn 20.1.0
backend_1  | [2023-03-27 11:23:57 +0000] [1] [INFO] Listening at: http://0.0.0.0:80␣
↪(1)
backend_1  | [2023-03-27 11:23:57 +0000] [1] [INFO] Using worker: gevent
backend_1  | [2023-03-27 11:23:57 +0000] [7] [INFO] Booting worker with pid: 7
backend_1  | [2023-03-27 11:23:57 +0000] [8] [INFO] Booting worker with pid: 8
backend_1  | [2023-03-27 11:23:57 +0000] [9] [INFO] Booting worker with pid: 9
backend_1  | [2023-03-27 11:23:57 +0000] [10] [INFO] Booting worker with pid: 10
backend_1  | [2023-03-27 11:23:57 +0000] [11] [INFO] Booting worker with pid: 11
backend_1  | [2023-03-27 11:23:57 +0000] [12] [INFO] Booting worker with pid: 12
backend_1  | [2023-03-27 11:23:57 +0000] [13] [INFO] Booting worker with pid: 13
backend_1  | [2023-03-27 11:23:57 +0000] [14] [INFO] Booting worker with pid: 14
backend_1  | INFO:root: * TissUUmaps version: 3.1.0.1
backend_1  | INFO:root: * TissUUmaps version: 3.1.0.1
backend_1  | INFO:root: * TissUUmaps version: 3.1.0.1
backend_1  | INFO:root: * TissUUmaps version: 3.1.0.1
backend_1  | INFO:root: * TissUUmaps version: 3.1.0.1
backend_1  | INFO:root: * TissUUmaps version: 3.1.0.1
backend_1  | INFO:root: * TissUUmaps version: 3.1.0.1
backend_1  | INFO:root: * TissUUmaps version: 3.1.0.1
```

5. Enter http://localhost:8050 in a browser to see TissUUmaps application running.

3.2.3 Configure sftp multi-user access (Optional)

Add the following lines to `/etc/ssh/sshd_config`:

```
Subsystem sftp internal-sftp
Match Group sftpusers
    ChrootDirectory /jail/shared/%u
    X11Forwarding no
    AllowTcpForwarding no
    ForceCommand internal-sftp
    PasswordAuthentication yes
```

Use the following bash script to create new sftp users **(as root)**:

```
#! /bin/bash

read -p 'Username: ' uservar
# Use the path mounted in docker:
MOUNT_TMAP='/jail'

# Create sftpusers group and $uservar user
groupadd sftpusers
adduser --gecos "" --home $MOUNT_TMAP/shared/$uservar $uservar
usermod -G sftpusers $uservar

# Create home for user in $MOUNT_TMAP/shared/ with correct permissions
mkdir -p $MOUNT_TMAP/shared/$uservar/files
chmod 755 $MOUNT_TMAP/shared/$uservar/
chown root:root $MOUNT_TMAP/shared/$uservar/
chmod 750 $MOUNT_TMAP/shared/$uservar/files/
chown $uservar:$uservar $MOUNT_TMAP/shared/$uservar/files

# Make sure all files created in the files folder are writable by $uservar
setfacl -m d:u:$uservar:rwx $MOUNT_TMAP/shared/$uservar/files/

echo 'To access files placed in the /files sftp folder, use:'
echo '    http://TMAP_URL/FILENAME?path='$uservar'/files'
```

The new users will be able to connect through sftp, and only see the files in `/jail/shared/USER_NAME/files`. Files will be available on the web using `http://localhost:8050/MYFILE.tmap?path=/USER_NAME/files`

Restart the ssh daemon with `sudo service ssh restart`.

3.2.4 Password protection of files hosted on TissUUmaps docker

You can add login and password for any folder hosted on the TissUUmaps server by adding a `auth` file containing your login and password in the format:

```
login:password
```

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

4.4 The TMAP file format

The TMAP format contains a description of image layers, markers, regions, and settings. It is highly recommended to create .tmap files by saving projects from TissUUmaps, but you can also edit the files manually to add or change projects' settings, or generate them as exported data from other software for import in TissUUmaps.

The TMAP format uses JSON, with the following specifications:

4.4.1 TMAP project specifications

Description of image layers, markers, regions, and settings of a project. 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		

continues on next page

Table 1 – continued from previous page

	• ^[0-9]+$	*LayerFilter*	
• filters	List of filters shown as active filters in the GUI under the Image layers tab		
	type	*array*	
	default	[“Saturation”, “Brightness”, “Contrast”]	
	items		
	•	*Filter*	
• compositeMode	Mode defining how image layers will be merged (composited) with each other. Valid string values are “source-over” and “lighter”, which correspond to ‘Channels’ and ‘Composite’ in the GUI.		
	type	*string*	
	default	source-over	
• mpp	The image scale in Microns Per Pixels. If not null, then adds a scale bar to the viewer. Set to 0 to display the scale bar in pixels.		
	type	*float*	
	default	null	
• boundingBox	Bounding box used to set initial zoom and pan on the view when loading the project.		
	type	*object*	
	default	null	
	properties		
	• **x**	Left coordinate of the bounding box in pixels	
		type	*float*
	• **y**	Top coordinate of the bounding box in pixels	
		type	*float*
	• **width**	Width of the bounding box in pixels	
		type	*float*
	• **height**	Height of the bounding box in pixels	
		type	*float*
• rotate	Angle of rotation of the view in degrees. Only multiples of 90 degrees are supported.		
	type	*integer*	
	default	0	
• markerFiles	type	*array*	
	default	[]	
	items		
	•	*MarkerFile*	
• regions	GeoJSON object, see *Regions section*.		
	type	*object*	
	default	{ }	
• regionFile	**(Deprecated)** GeoJSON region file loaded on project initialization. Use regionFiles instead.		
	type	*string*	
	default		
• regionFiles	type	*array*	
	default	[]	
	items		

continues on next page

Table 1 – continued from previous page

	•	*RegionFile*	
• plugins	List of plugins to load with the project. See also the *Plugins section*.		
	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

Description of an image layer. Required properties are shown in **bold** text		
type	*object*	
properties		
• **name**	Name of the image layer	
	type	*string*
• **tileSource**	Relative path to an image file in a supported format. See also the *Images section*.	
	type	*string*

LayerFilter

Description of an image filter to be applied to the pixels in an image layer. Required properties are shown in **bold** text			
type	*array*		
items			
•	type	*object*	
	properties		
	• **name**	Filter name. See *Filter* for more details.	
		type	*string*
	• **value**	Filter parameter. See *Filter* for more details.	
		type	*string*

Filter

TissUUmaps supports most filters available in OpenSeadragon via the https://github.com/usnistgov/OpenSeadragonFiltering plugin.	
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' are 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
• hideSettings	Hide markers' settings and add a toggle button instead.	
	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. If array of string, then a dropdown is created instead of a button.	
	type	*string / array*
• 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*

continues on next page

Table 2 – continued from previous page

	default	null
• gb_name	Name of CSV column to display for groups instead of group key value	
	type	*string*
	default	null
• 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. TissUUmaps expects labels and numerical values for sectors to be separated by ':' characters in the CSV column data.	
	type	*string*
	default	null
• pie_dict	JSON string specifying a custom dictionary for mapping pie chart sector indices to colors. Example: `'{0: "#ff0000", 1: "#00ff00", 2: "#0000ff"}'`. If no dictionary is specified, TissUUmaps will use a default color palette instead.	
	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	

continues on next page

Table 2 – continued from previous page

	type	*string*
	default	1
• tooltip_fmt	Custom formatting string used for displaying metadata about a selected marker. See https://github.com/TissUUmaps/TissUUmaps/issues/2 for an overview of the grammer and keywords. If no string is specified, TissUUmaps will show default metadata depending on the context.	
	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

continues on next page

Table 3 – continued from previous page

• 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

RegionFile

Description of settings and regions loaded from GeoJSON file. Required properties are shown in **bold** text.		
type	*object*	
properties		
• **title**	Name of region button	
	type	*string*
• comment	Optional description text shown next to region button	
	type	*string*
	default	
• autoLoad	If the regions should be automatically loaded when the TMAP project is opened. If this is false, the user instead has to click on the region button in the GUI to load the regions.	
	type	*boolean*
	default	false
• **path**	Relative file path to GeoJSON file in which marker data is stored. If array of string, then a dropdown is created instead of a button.	
	type	*string / array*
• settings	type	*array*
	default	[]
	items	
	•	*Setting*

Setting

Advanced javascript setting, used to change the value of a property in a module, or to call a function in a module. If module.function is a function, then calls `module.function(value)`, else affects `value` to `module.function`. Required properties are shown in **bold** text.		
type	*object*	
properties		
• **module**	Module where the function or property lies.	
	type	*string*
• **function**	Function or property of the given module.	
	type	*string*
• **value**	type	*number*

4.4.2 Example of a .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)

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

CHAPTER

FIVE

SUPPORT

5.1 How to seek support and discussion

If you want to ask questions about TissUUmaps please visit forum.image.sc.

5.2 How to issue an error on GitHub

If you want to report a bug, please do so at issues on GitHub.

INDEX