

Sep 17, 2024

Version 2

QUINT Workflow for Fluorescence V.2

DOI

<https://dx.doi.org/10.17504/protocols.io.4r3l22y6jl1y/v2>

Michael X. Henderson¹

¹Van Andel institute

Michael X. Henderson: ORCID: 0000-0001-9710-0726



Michael Henderson

Van Andel Research Institute

Create & collaborate more with a free account

Edit and publish protocols, collaborate in communities, share insights through comments, and track progress with run records.

Create free account

OPEN  ACCESS



DOI: <https://dx.doi.org/10.17504/protocols.io.4r3l22y6jl1y/v2>

Protocol Citation: Michael X. Henderson 2024. QUINT Workflow for Fluorescence. [protocols.io](https://dx.doi.org/10.17504/protocols.io.4r3l22y6jl1y/v2)
<https://dx.doi.org/10.17504/protocols.io.4r3l22y6jl1y/v2> Version created by [Lindsay Meyerdirk](#)

License: This is an open access protocol distributed under the terms of the [Creative Commons Attribution License](#), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited

Protocol status: Working

We use this protocol and it's working



Created: September 12, 2024

Last Modified: September 17, 2024

Protocol Integer ID: 107483

Keywords: ASAPCRN, quint workflow for fluorescence, quint workflow, published quint workflow, fluorescence, quint, workflow, pmid

Funders Acknowledgements:

Aligning Science Across Parkinson's

Grant ID: ASAP-020616

National Institute on Aging

Grant ID: R01-AG077573

Abstract

This protocol describes QUINT workflow for fluorescence. It was updated 9/17/2024. It is based on the published QUINT workflow established by Yates and colleagues (PMID: 31849633).

Attachments



[812-2118.pdf](#)

4.6MB



Guidelines

Purpose

The purpose of this workflow is to enable mouse brain segmentation, registration, and quantification of regional signals. The simplest segmentation is done in QuPath because this program handles whole slide images and has good segmentation algorithms. Registration is done using 3 programs: QuickNII (aligns to a 3D atlas, typically the 2017 CCFv3 Allen Brain Atlas), VisuAlign (allows for non-linear warp transformation of the atlas to match sections), QMask (masks each side of the brain to allow for bilateral assessment of brain regions). Segmentations and registrations are then brought together in Nutil, enabling the generation of quantitative measures for every region of the brain.

Necessary Programs and Locations

1. QuPath: <https://qupath.github.io/>
2. DeepSlice: <https://www.deepslice.com.au/>
3. QuickNII: <https://www.nitrc.org/projects/quicknii> (SELECT ABA Mouse Edition)
4. VisuAlign: <https://www.nitrc.org/projects/visualign/>
5. QMask: https://www.nitrc.org/frs/?group_id=1341
 - a. Listed as QuickMask.zip
6. Nutil: <https://www.nitrc.org/projects/nutil/>
 - *QuickNII, VisuAlign, and Nutil can be a bit finicky, and we have found it is best to have these folders on the desktop.

Scanned Slides

To enable high throughput, slides are scanned on the Axioscan microscope. After slide scanning, images will be found at \\pn.vai.org\projects_primary\henderson\vari-core-generated-data\Axioscan\RNAScope. Move the slides into either the Human or Mouse folder and then into its corresponding stain and channel folder. If a folder does not exist, create one following the naming convention (i.e., Stain3_Slc17a7_Gad1_pS129).

Folder Organization

The programs for registration and quantification rely on having exact file paths from which to call the data. Therefore, it is easiest to set these folders up from the beginning and using the following layout:

1. QUINT Workflow
 - a. slide#stain#



Note

*This folder must be empty when you select it for your QuPath project in step #2 under the QuPath Visualization/Segmentation section. Once that is completed, you can then continue to create the following subfolders.

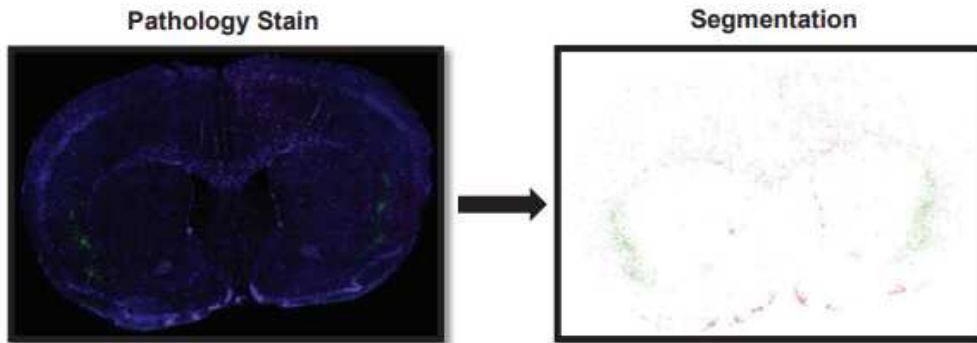
- i. QVN (for QuickNII, VisuAlign, Nutil)
 - 1. Atlas
 - 2. Input
 - 3. Mask
 - 4. Output_Left
 - 5. Output_Right

QuPath Visualization/Segmentation

1

Note

QuPath is a visualization and segmentation platform optimized for whole slide images. This is the place where you can visualize all your slides and create the output that will be used for all other programs.



Open the **QuPath** application. Select 'Create project'.

2

3

QuPath Visualization/Segmentation

- 4 Select your slide/stain folder within your QUINT Workflow project folder (i.e., slide24stain19).
- 5 Select 'Add images' > 'Choose files'. Navigate to the image file and select 'Open'. Set image type to Fluorescence. Select 'Import'.
- 6 Double click on the first image for it to appear in the viewing window.
- 7 Open the Brightness & contrast window

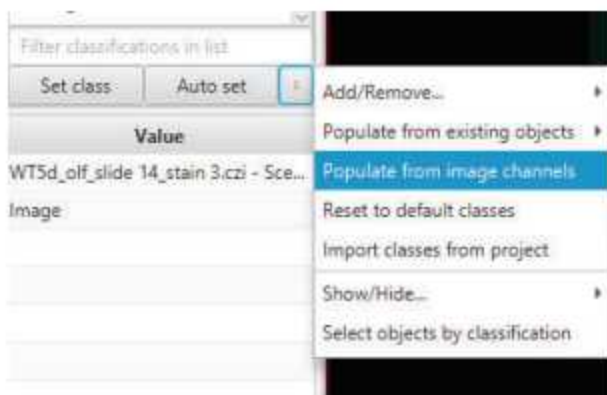


and leave it open.

- 8 Select 'Automate' > 'Show script editor'. Run the following script (change the channel names to the corresponding channels for your stain). This will change your channel names to the appropriate names.

```
1 setChannelNames (  
2 'DAPI',  
3 'Nr4a2',  
4 'ps129'  
5 )
```

- 9 Stay on the first image. In the script editor window select 'Run' > 'Run for project'. Move all the images over to the selected column except for the first image. You cannot run a script for the whole project on an image that you have currently open in the viewer. Select 'OK'. Close the script editor.
- 10 For each image, adjust the Min and Max display in the Brightness & contrast window for each channel as needed to make the scan easier to see. This only changes your visual, no actual properties of the scan. Pathology Stain Segmentation
- 11 Create annotation classifications for any new channel names you have previously not worked with. Select the 'Annotations' tab > three dots in the bottom right of the window > populate from image channels. Ensure that the class color matches the channel color.



QuPath Visualization/Segmentation: Image Export for Registration

- 12 Select the rectangle button



and select your ROI around the first brain.

- 13 Lock the rectangle by selecting the annotation (the rectangle will appear yellow). Right click inside the rectangle, select 'Annotations' > 'Lock'. This will lock the rectangle, so it is not accidentally moved.
- 14 MAKE SURE THE RECTANGLE IS SELECTED (YELLOW) AS YOU COMPLETE THIS STEP. Select 'File' > 'Export Images' > 'Rendered RBG (with overlays)'. Set the export format to (PNG). Set the downsample factor to 12.0. Select OK.
- 15 Save in the QVN folder with the appropriate naming designation (i.e., sl18st3sc1_s041, sl18st3sc2_s042_).
- 16 Repeat steps 10-13 for each image.

QuPath Visualization/Segmentation: Cell Detection

- 17 Select 'Analyze' > 'Cell Detection' > 'Cell Detection' to open the cell detection window.
- 18 Adjust the Cell Detection parameters as needed. The most helpful parameters to adjust are Sigma, Minimum Area, Maximum Area, Threshold, and Cell Expansion.
 - a. Minimum Area and Maximum Area will be dependent on whether you are analyzing mouse or human tissue.
 - b. Keep all checkboxes checked "Split by Shape", "Include Cell Nucleus", "Smooth Boundaries", and "Make Measurements".
 - c. A lower Sigma value will break up nuclei more. A higher Sigma value will combine nuclei more often.
 - d. Once you are satisfied with the parameters, you can save the Cell Detection script to easily apply it to your other images ('Workflow' tab → Create script). The same cell detection parameters must be applied to all images.
- 19 Remove cells from areas of damaged tissue or outside the brain.

a. Select scrub tool



and select button



and drag over areas with cells to remove. Once the right cells are highlighted, hit backspace and delete. Turn off select button and select move tool



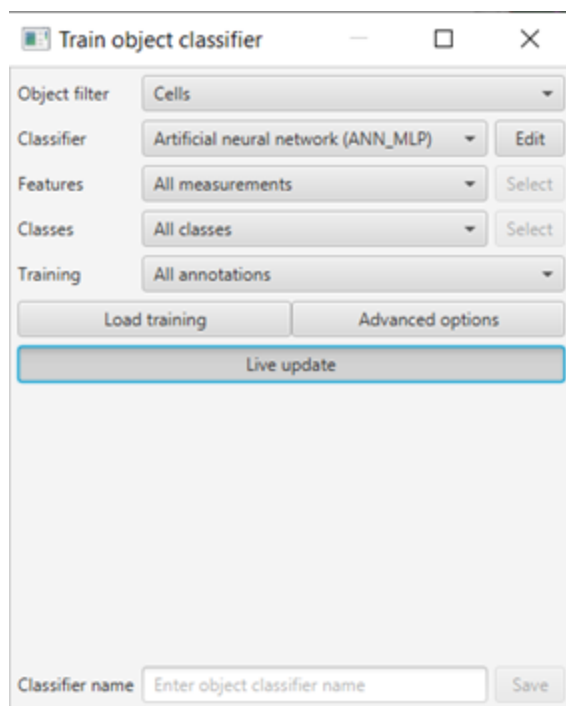
to return to normal when finished.

b. Alternatively, Ctrl+click individual cells and delete with backspace.

QuPath Visualization/Segmentation: Training Object Classifiers

- 20 Select the three dots to the right of 'Auto Set' > 'Add/Remove...' > 'Add class'. Name the new class 'Training'.
- 21 Choose 6-7 regions across the image set to use as training images. These regions should be ones that contain a variety of staining to properly train the classifier. Take regions from more than one section get a broad range of what your positive cells look like (like 1-2 regions per image).
- 22 For each annotation rectangle created for the training classifier, set its class to 'Training' by right clicking in the annotation rectangle > 'Set class' > 'Training'. Save (File/Save or Ctrl+S) after making final annotation.
- 23 Once you have created all your training annotations, select 'Classify' > 'Training images' > 'Create training image'. Set the classification of your combined training annotations to 'Training' and select 'Rectangles only'. Set the image type of your combined training annotations to 'Fluorescence'.

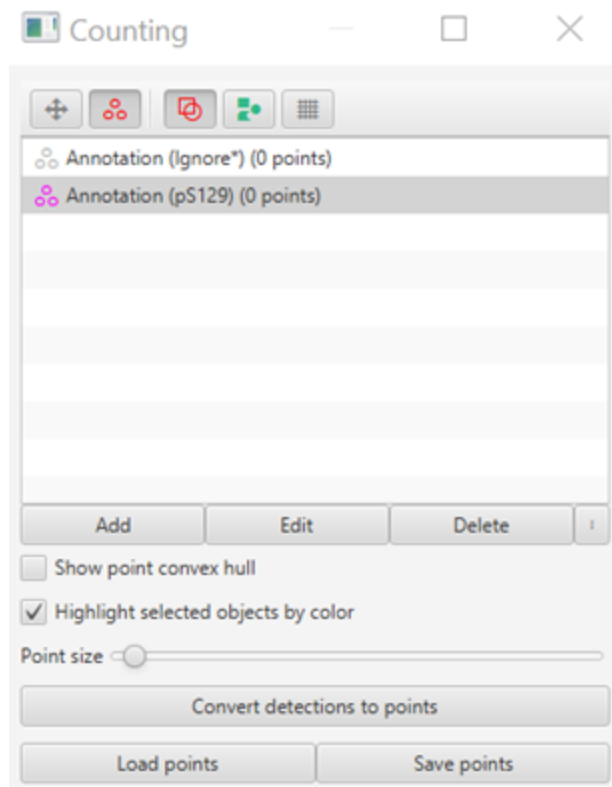
- 24 Rename training image (right click on image> rename image) to the channel name you will train the classifier to detect. Create a duplicate of the combined training annotations for each different channel you have (excluding your nuclear stain). Rename the image to include the channel name. To duplicate, right click on the combined training annotation image and select 'Duplicate image(s)'.
- 25 Draw a rectangle annotation around each training image. Lock the annotation. Run cell detection with the same parameters as your individual images.
- 26 Select 'Classify' > 'Object classification' > 'Train object classifier'. Set the object filter to 'Cells', the classifier to 'Artificial neural network (ANN_MLP)', features to 'Selected measurements', classes to 'All classes', and training to 'All annotations'. Select 'Select' to the right of 'Selected measurements' and select all measurements corresponding to the selected training channel (input channel name in Filter type box and hit select all, apply). Leave the window open.



- 27 Open the points window



Select 'Add' two times to add two annotations. Set the first annotation to 'Ignore*' by right clicking on the annotation and select 'Set class' > 'Ignore*'. Set the second annotation to the channel you are training.

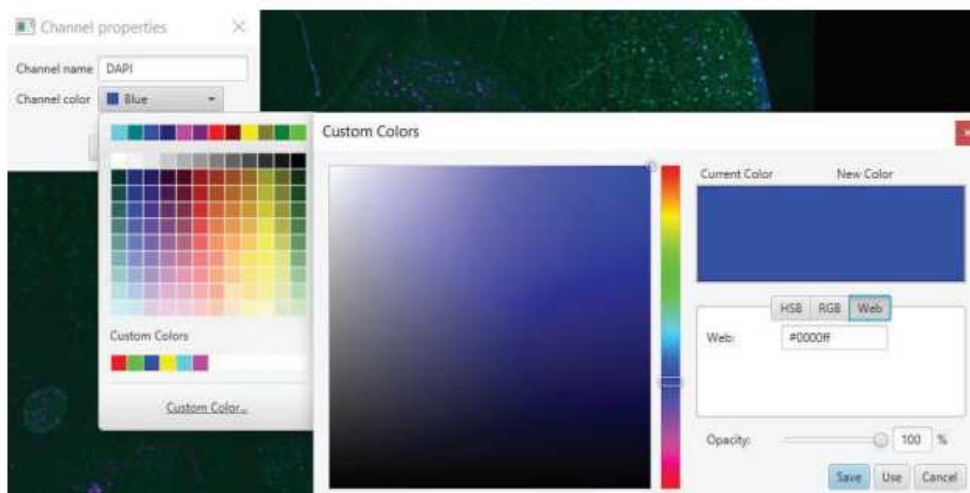


- 28 In the Brightness & contrast window, turn off all channels except the channel you are training. To better visualize stain, hide detected cells by clicking detection objects button



highlight channel of interest and play with channel min/max to optimize stain and eliminate background, show cells by using same button as before.

- 29 Train the classifier by using the points annotations to identify cells as either positive for the channel or ignore. Continue training the classifier until you are satisfied with the predicted output. Select live update to view how the points you select are affecting the classifier(place at least 1 point for each class before viewing live update).
- 30 Give the classifier a name with the slide number, stain number, the channel, and the date (i.e., slide24stain19_Nr4a2_12.16.22). Select 'Save'.
- 31 Repeat steps 24-28 for every training image.
- 32 Select 'Classify' > 'Object classification' > 'Create composite classifier'. Move the individual classifiers to the 'selected' side of the window in the order you want them to be applied. Enter a name for the combined classifier (i.e., sl18st19_composite_11.22.22) and select 'Save & apply'.
- 33 For each whole brain image, select 'Classify' > 'Object classification > 'Load object classifier', select the composite classifier you just created and 'Apply classifier'.
- 34 To show sub-classes identified from your composite classifier (cells identified as positive for multiple channels) navigate to the 'Annotations' tab > three dots button in the bottom right of the window > 'Populate from existing objects' > 'All classes (including sub-classes)'. Select 'Yes' when asked if you want to keep the existing available classes.
- 35 Write down the 6-digit web color code for each sub-class in your annotation window. You will need this information for Nutil. Double click the channels listed in the annotations tab to view the color code.



QuPath Visualization/Segmentation: Export Segmentation

- 36 Navigate to each scene image and rename the annotation that is off the entire brain image. Rename by right-clicking on the selected annotation image > 'Annotations' > 'Set properties'. Rename the annotation using the following naming convention: 'sl18st19sc1input_s041_' ; 'sl18st19sc2input_s042_'.
- 37 Scenes with smaller annotation regions that you used to make your training images will have a second segmentation. Delete the segmentation image that corresponds to the smaller annotation,
- 38 Navigate to one of your combined training images.
- 39 Select 'Automate' > 'Show script editor'.
- 40 Enter the script in the blue box below at line 1. You can export multiple cell classifications together to simplify the downstream Nutil input. To do so, add more lines with

```
13 .addLabel('pS129', 1)
14 .addLabel('Nr4a2', 2)
15 .addLabel('pS129:Nr4a2', 3)|
```

".addLabel('Name of classification', #).
- 41 Select 'Run' > 'Run for project'. Move each scene image over to the 'Selected' window and select 'OK'.

```
import qupath.lib.images.servers.LabeledImageServer
def imageData = getCurrentImageData()
// Define output path (relative to project)
def name =
GeneralTools.getNameWithoutExtension(imageData.getServer().getMeta
data().getName())
def pathOutput = buildFilePath(PROJECT_BASE_DIR, 'export', name)
mkdirs(pathOutput)
// Export at full resolution
double downsample = 1.0
// Create an ImageServer where the pixels are derived from
annotations
def labelServer = new LabeledImageServer.Builder(imageData)
.backgroundLabel(0, ColorTools.WHITE) // Specify background label
(usually 0 or 255)
.downsample(downsample) // Choose server resolution; this should
match the resolution at which tiles are exported
.addLabel('Slc17a7', 1) // Choose output labels (the order
matters!)
.addLabel('Gad1', 2)
.useCellNuclei()
.multichannelOutput(false) // If true, each label refers to the
channel of a multichannel binary image (required for multiclass
probability)
.build()
// Export each region
int i = 0
for (annotation in getAnnotationObjects()) {
name = annotation.getName()
if (annotation.getROI().getRoiName() == "Rectangle") {
def region = RegionRequest.createInstance(
labelServer.getPath(), downsample, annotation.getROI())
i++
def outputPath = buildFilePath(pathOutput, name + '.png')
writeImageRegion(labelServer, region, outputPath)
}}
```

Command

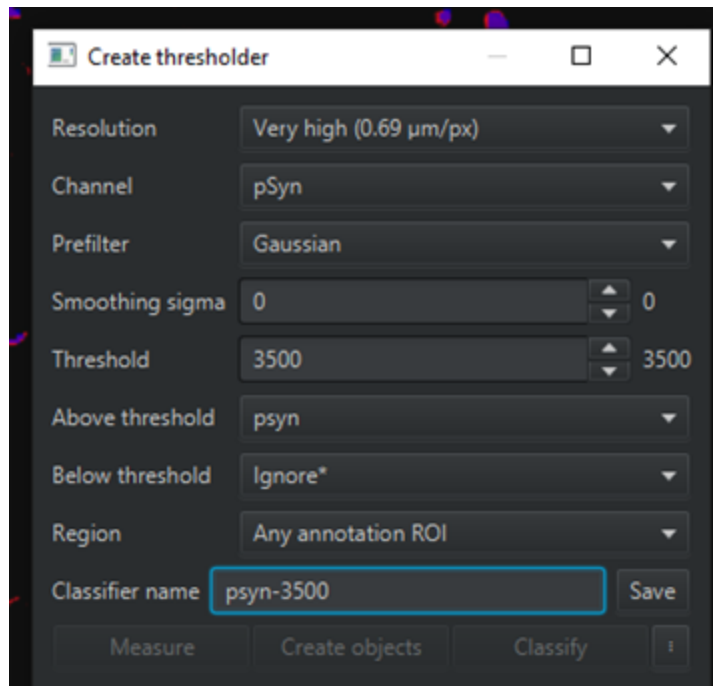
new command name

```
import qupath.lib.images.servers.LabeledImageServer
def imageData = getCurrentImageData()
// Define output path (relative to project)
def name =
GeneralTools.getNameWithoutExtension(imageData.getServer().getMetadata
().getName())
def pathOutput = buildFilePath(PROJECT_BASE_DIR, 'export', name)
mkdirs(pathOutput)
// Export at full resolution
double downsample = 1.0
// Create an ImageServer where the pixels are derived from annotations
def labelServer = new LabeledImageServer.Builder(imageData)
    .backgroundLabel(0, ColorTools.WHITE) // Specify background label
(usually 0 or 255)
    .downsample(downsample) // Choose server resolution; this should
match the resolution at which tiles are exported
    .addLabel('Slc17a7', 1) // Choose output labels (the order matters!)
    .addLabel('Gad1', 2)
    .useCellNuclei()
    .multichannelOutput(false) // If true, each label refers to the
channel of a multichannel binary image (required for multiclass
probability)
    .build()
// Export each region
int i = 0
for (annotation in getAnnotationObjects()) {
name = annotation.getName()
if (annotation.getROI().getRoiName() == "Rectangle") {
        def region = RegionRequest.createInstance(
labelServer.getPath(), downsample, annotation.getROI())
        i++
def outputPath = buildFilePath(pathOutput, name + '.png')
        writeImageRegion(labelServer, region, outputPath)
}}
```

- 42 Find the exported images in the export folder. The segmentation image should be white with objects in their WEB ID color. The images should be the exact same dimensions of the earlier region of interest image you created as a PNG with a downsize factor of 12.
- 43 Scenes with smaller annotation regions that you used to make your training images will have a second segmentation. Delete the segmentation image that corresponds to the smaller annotation.
- 44 Copy the images to the 'Input' folder in the QVN folder. These images will be used in the final step of Nutil. Close the QuPath application.

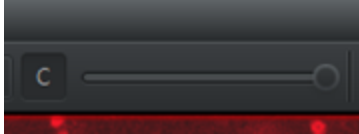
QuPath Visualization/Segmentation: Alternate Segmentations, Pixel Classifier for Florescence

- 45 Isolate the image display to show only the channel you want to segment for. Optimize stain visualization by playing with display min/max.
- 46 Create threshold: Classify>Pixel Classifier>Create Threshold



- a. Set Resolution: We usually use Very High (.69um/px) (Note Resolution for Nutil processing later in workflow).
- b. Set Channel to channel of interest
- c. Leave Prefilter as Gaussian and Smoothing sigma at 0

- d.** Before playing with the threshold, set Above threshold to the classification that works for you. This can be Positive from the default classes, or you can make your own (3 dots next to auto set in Classes tab>Add/Remove>Add Class –edit color by double clicking class in classes window). Set Below threshold to Ignore*.
- e.** Optimize threshold – this value is based on fluorescence intensity as seen in display min/max - usually in the thousands range. Ensure the classifier is only detecting positive signal and adjust threshold number as necessary.
- i.** You can utilize the C Classifier button to turn on/off the classifier and scrollbar to adjust opacity of detected signal.



- f.** Give your classifier a name and Create objects to apply it to the open image

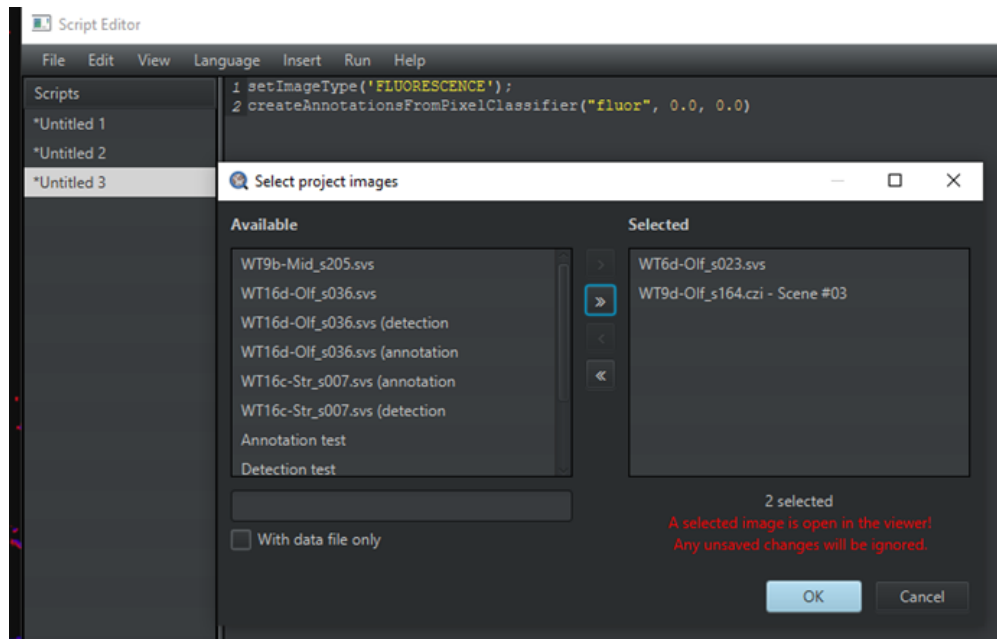
47

You can apply the classifier you just made to other images by loading the classifier when other images are open: *Classify>Pixel Classifier>Load pixel classifier>Choose model>Select made classifier.*

- a.** You can also generate a workflow by selecting 'Create script' at the bottom of the workflow tab. This will turn all the commands executed on the open image into Groovy script. You can batch run this script for other images in your project by going to Run>Run for project and selecting for desired images

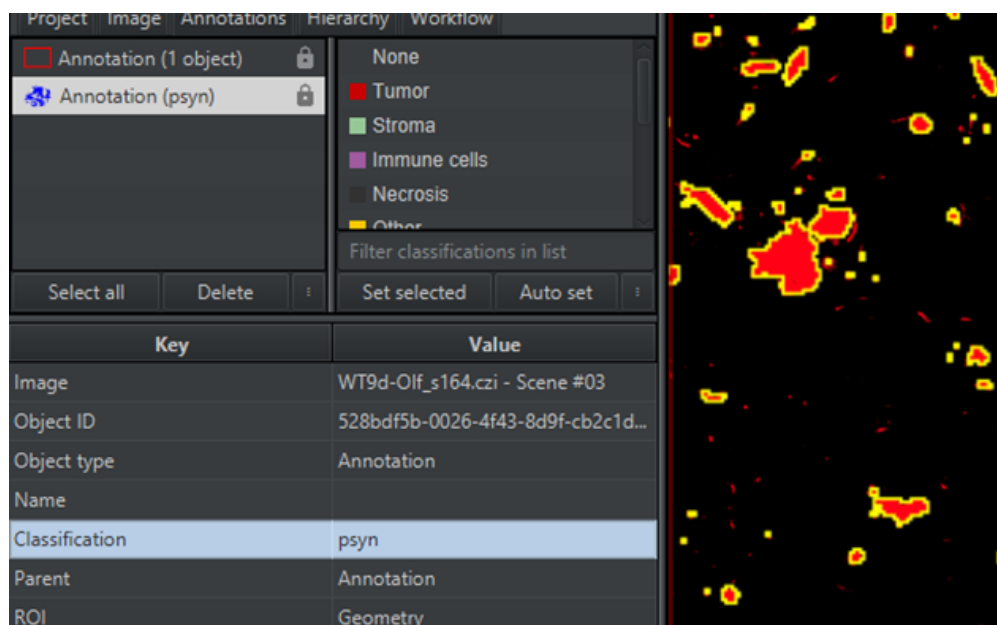
Note

After running script, be sure to save changes to the image you have opened before switching off images to avoid losing work



48 Export segmentation image using tweaked script from brightfield workflow below.

a. Note in line 13, (.addLabel('Insert class here', 1)) includes the name of the class given to the annotation generated by your classifier. This can be found by selecting the annotation and locating Classification in the Measurements box underneath it.





```
import qupath.lib.images.servers.LabeledImageServer

def imageData = getCurrentImageData()

// Define output path (relative to project)
def name = GeneralTools.getNameWithoutExtension(imageData.getServer().getMetadata().getName())
def pathOutput = buildFilePath(PROJECT_BASE_DIR, 'export', name)

mkdirs(pathOutput)

// Export at full resolution
double downsample = 1.0

// Create an ImageServer where the pixels are derived from annotations
def labelServer = new LabeledImageServer.Builder(imageData)

.backgroundLabel(0, ColorTools.WHITE) // Specify background label (usually 0 or 255)

.downsample(downsample) // Choose server resolution; this should match the resolution at which tiles are exported

.addLabel('psyn', 1) // Choose output labels (the order matters!)

.multichannelOutput(false) // If true, each label refers to the channel of a multichannel binary image (required for multiclass probability)

.build()

// Export each region
int i = 0
for (annotation in getAnnotationObjects()) {
    if (annotation.getROI().getRoiName() == "Rectangle") {
        def region = RegionRequest.createInstance(
            labelServer.getPath(), downsample, annotation.getROI())

        i++

        def outputPath = buildFilePath(pathOutput, name + '.png')
        writeImageRegion(labelServer, region, outputPath)
    }
}
```

Command

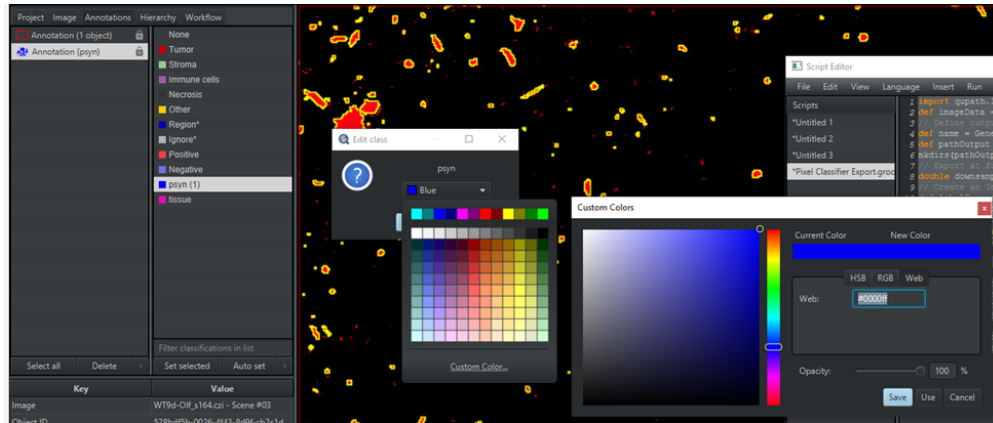
new command name

```
import qupath.lib.images.servers.LabeledImageServer
def imageData = getCurrentImageData()
// Define output path (relative to project)
def name =
GeneralTools.getNameWithoutExtension(imageData.getServer().getMetadata
().getName())
def pathOutput = buildFilePath(PROJECT_BASE_DIR, 'export', name)
mkdirs(pathOutput)
// Export at full resolution
double downsample = 1.0
// Create an ImageServer where the pixels are derived from annotations
def labelServer = new LabeledImageServer.Builder(imageData)
    .backgroundLabel(0, ColorTools.WHITE) // Specify background label
(usually 0 or 255)
    .downsample(downsample) // Choose server resolution; this should
match the resolution at which tiles are exported
    .addLabel('psyn', 1) // Choose output labels (the order matters!)
    .multichannelOutput(false) // If true, each label refers to the
channel of a multichannel binary image (required for multiclass
probability)
    .build()
// Export each region
int i = 0
for (annotation in getAnnotationObjects()) {
    if (annotation.getROI().getRoiName() == "Rectangle") {
        def region = RegionRequest.createInstance(
            labelServer.getPath(), downsample, annotation.getROI())
        i++
        def outputPath = buildFilePath(pathOutput, name + '.png')
        writeImageRegion(labelServer, region, outputPath)
    }
}
```

- 49 Find exported images in your folder scheme under QuPath\export. The images should be the

exact same dimensions of the earlier region of interest image you created as a PNG with a
downsize factor of 12.

a. The segmentation image should have a white background with segmented pixels in the color of the annotation class. (Nutil will ask you to insert the object color as a 6-digit alphanumeric web color code. This can be found by *double clicking your annotation's class in the Classes tab>Selecting Custom Color from Dropdown>Selecting Web*



50 Place the image into **"Input"** folder that will be used in final Nutil step.

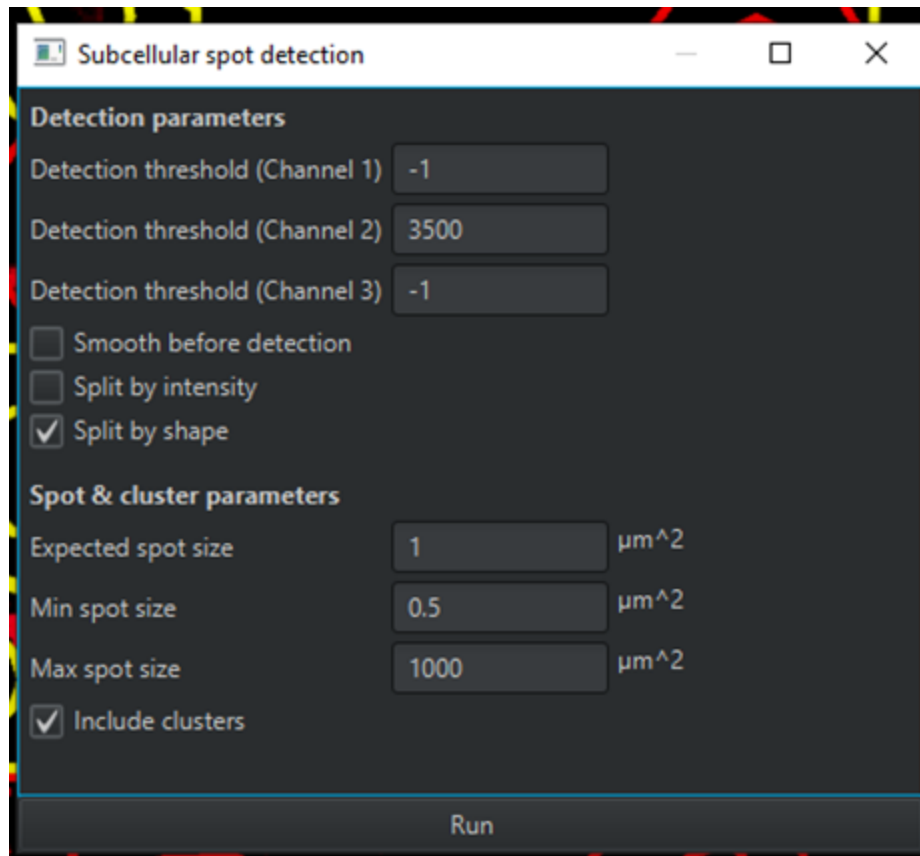
QuPath Visualization/Segmentation: Alternate Segmentations, Subcellular Segmentation

51

Note

Restricts pixel classifier to only within detected cells (inclusion area).

After running cell detection and object classifiers, open Subcellular spot detection window: Analyze>Cell Detection>Subcellular detection (experimental)



- 51.1 Set the detection threshold to the fluorescence value of your pixel classifier
- a. The order of the channels in the display window corresponds to the numbering in the subcellular spot detection window. Set the threshold in the channel number of interest and leave the other two at -1 to ignore.
 - b. Check split by shape to get individual stain objects within each cell
- 51.2 Adjust max spot size (adjust to something significantly larger than a single cell area)
- 52 To restrict subcellular detection to a specific cell type right click on class for desired cell type(s) > *Select objects by classification*
- a. Selected class cells should appear yellow with all other cells remaining their original color
 - b. Run the subcell detection command
(if you want to run the subcellular detection for all cells, this step is not necessary)

- 53 Export subcellular segmentation by running this script adapted from pixel classifier export script (note line 13* include the name of class of subcellular object → can be obtained by clicking 3 dots next to Auto set in Annotation Classes tab>*Populate from existing objects (All classes (including sub-classes))*):

```
import qupath.lib.images.servers.LabeledImageServer

def imageData = getCurrentImageData()

// Define output path (relative to project)
def name = GeneralTools.getNameWithoutExtension(imageData.getServer().getMetadata().getName())
def pathOutput = buildFilePath(PROJECT_BASE_DIR, 'export', name)
mkdirs(pathOutput)

// Export at full resolution
double downsample = 1.0

// Create an ImageServer where the pixels are derived from annotations
def labelServer = new LabeledImageServer.Builder(imageData)

    .backgroundLabel(0, ColorTools.WHITE) // Specify background label (usually 0 or 255)
    .downsample(1) // Choose server resolution; this should match the resolution at which tiles are exported
    .addLabel('Subcellular spot: Channel 2 object', 1)
    .multichannelOutput(false) // If true, each label refers to the channel of a multichannel binary image (required for multiclass probability)
    .useDetections()
    .build()

// Export each region
int i = 0
for (annotation in getAnnotationObjects()) {
    if (annotation.getROI().getRoiName() == "Rectangle") {
        def region = RegionRequest.createInstance(
            labelServer.getPath(), downsample, annotation.getROI()
        )
        i++
        def outputPath = buildFilePath(pathOutput, name + i + '.png')
        writeImageRegion(labelServer, region, outputPath)
    }
}
```

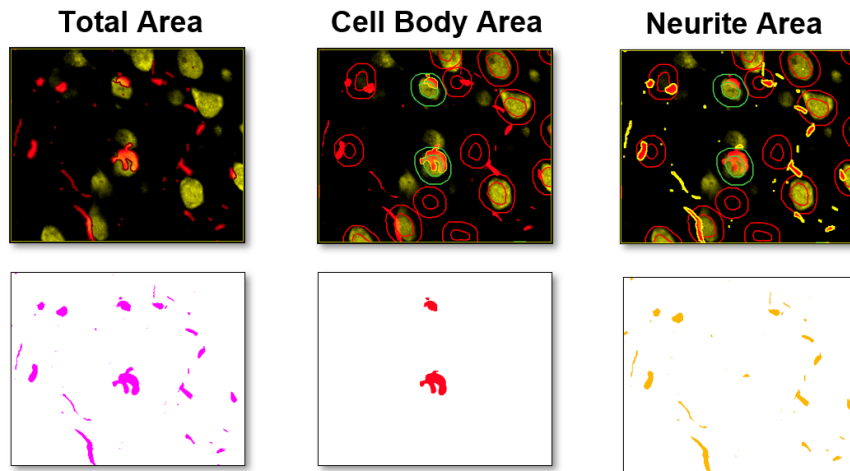
Command

new command name

```
import qupath.lib.images.servers.LabeledImageServer
def imageData = getCurrentImageData()
// Define output path (relative to project)
def name =
GeneralTools.getNameWithoutExtension(imageData.getServer().getMetadata
().getName())
def pathOutput = buildFilePath(PROJECT_BASE_DIR, 'export', name)
mkdirs(pathOutput)
// Export at full resolution
double downsample = 1.0
// Create an ImageServer where the pixels are derived from annotations
def labelServer = new LabeledImageServer.Builder(imageData)
    .backgroundLabel(0, ColorTools.WHITE) // Specify background label
(usually 0 or 255)
    .downsample(1) // Choose server resolution; this should match
the resolution at which tiles are exported
    .addLabel('Subcellular spot: Channel 2 object', 1)
    .multichannelOutput(false) // If true, each label refers to the
channel of a multichannel binary image (required for multiclass
probability)
    .useDetections()
    .build()
// Export each region
int i = 0
for (annotation in getAnnotationObjects()) {
    if (annotation.getROI().getRoiName() == "Rectangle") {
        def region = RegionRequest.createInstance(
            labelServer.getPath(), downsample, annotation.getROI())
        i++
        def outputPath = buildFilePath(pathOutput, name + '.png')
        writeImageRegion(labelServer, region, outputPath)
    }
}
```

The same principles for file location in fluorescent pixel classifier export applies here.

- 54 **Neurite Area:** Using the pixel classifier export that detects positive area signal everywhere, and the subcellular detection that detects only in cells, we can get neurite area (positive area outside of cells) by using the post-processing tool, Nutil2Usable. Subtracting subcellular area from total area will leave us with only area outside of cells (aka neurite). There is a protocol in N2U that instructs on how to go about that analysis.

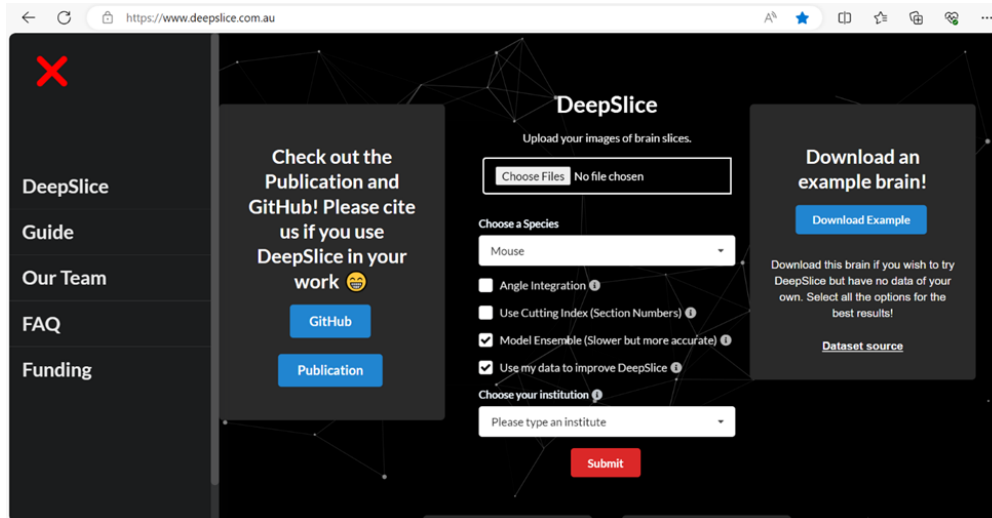


DeepSlice Automated Alignment

55

Note

DeepSlice is a deep neural network that automatically aligns mouse histology images through the Allen Brain Atlas coordinate framework. Alignments are viewable and refinable in QuickNII and set up sections to give a good starting point near and around each section's correct plane. DeepSlice's alignment is not completely accurate and further fine tuning in QuickNII is necessary. This is a newly developed tool and not imperative for the workflow but helps speed the process of registration and substitutes the Filebuilder step.



Open **DeepSlice** in your web browser.

56 Select *Choose Files*.

a. Upload all images for registration from the QVN folder

57 Ensure *Mouse* is selected for species, and *Model Ensemble* is checked (uses two DeepSlice versions to optimize alignment).

a. Avoid checking *Angle Integration* (this aligns all your brain sections to the same angle, which is inaccurate when blocks have different cutting angles)

b. Avoid checking *Using Cutting Index* (this suggests your sections numbers (_s###) correspond to serial section numbers spaced equally apart)

c. Optional to allow DeepSlice to use your data to improve the neural network and its predictive accuracy.

58 Select **Submit**.

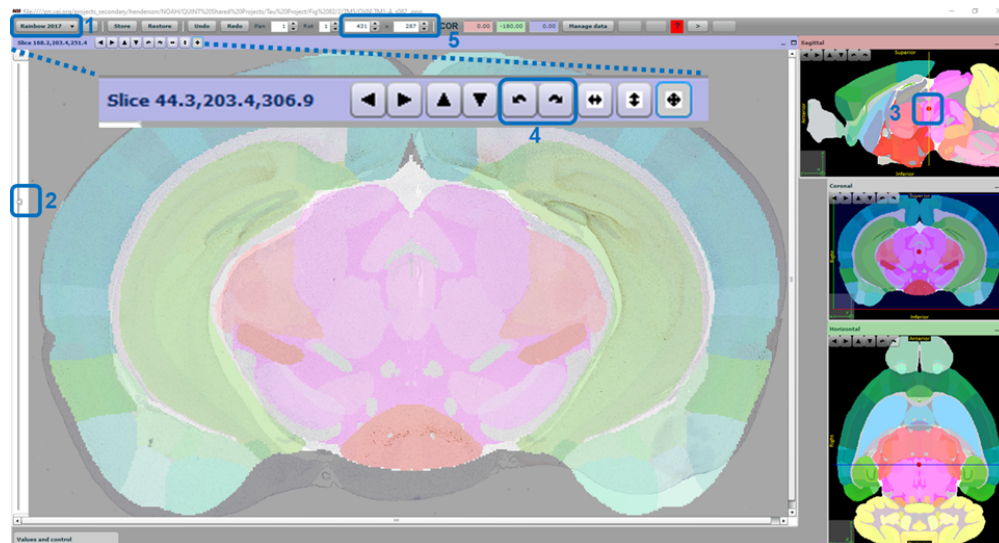
59 After all sections/slices are processed, press Download XML

60 Insert the downloaded XML (titled by default 'results.XML') into your QVN folder with all your images for registration.



QuickNII Brain Atlas Registration

- 61 Open the **QuickNII** program folder.
- 62 Open **Filebuilder**. (FileBuilder loads your registration images by default settings without automated alignment, Skip to *step 68* if using **DeepSlice**).
- 63 Navigate to the **QVN** folder with the brain image exports from QuPath. These images are not the segmentation exports, but the original brain image exports. These images must all be surrounded by a yellow rectangle.
- 64 Select all the images to be registered and select 'Open'. *It is useful to add a shortcut of your QUINT Workflow folder to your desktop for simpler navigation.*
- 65 Select 'Save XML'. Navigate to the QVN folder and save as 'Filebuilder XML'. Make sure to save this file in the same folder as the brain image exports from QuPath.
- 66 Close Filebuilder.
- 67 Open the QuickNII application. Select 'Manage data' > 'Load' and select the XML file that was just generated in step 66 (**DeepSlice**: load results.XML from QVN folder).
- 68 Double click on the first image in Filebuilder to have it show up in the viewing window in the QuickNII application.



- 69 Select 'Rainbow 2017' from the drop-down menu in the upper left-hand corner of the toolbar (1).
- 70 You can adjust how you see the atlas overlay by dragging the vertical transparency bar on the left side of the screen (2).
- 71 For the first section, find the anteroposterior position. To do this, drag the sagittal red dot (3) to the correct rostro-caudal position. Select 'Store' to save the position.
- 72 Repeat step 72 for the last section. This will bring all other sections to the approximately correct position.
- 73 Adjust each individual section to the appropriate place in the atlas. Many adjustments may need to be made until the correct plane of section is identified.

Note

NOTE: All the atlas needs to remain in view, or it will be lost for analysis. Keep the atlas image smaller than the brain image. All of the Atlas needs to remain in the viewer or it will be lost for analysis. *Alignment will not be perfect, only the plane of section, but the better job you do here, the easier VisuAlign transformations will be.*

- Rotation:** clockwise or counterclockwise (4).
- Brain size:** in the x and y direction (5).
- Rostro-caudal position:** adjust sagittal view.

d. **Left-right plane:** adjust horizontal view.

e. **Front-back plane:** adjust sagittal view.

74 Select 'Store' before moving off a section or it will not save!



75 Navigate to the next section by double clicking on the section in the Filebuilder, or by selecting the < and > arrows in the upper-right corner. Edit all sections as noted in step 13

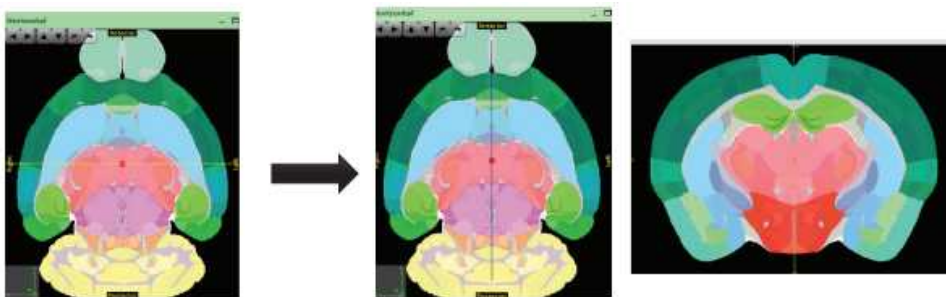
76 Select 'Manage data' > 'Export Propagation' and save this XML file as "QuickN XML.xml" within the QVN folder. *It is not automatically recognized as a .xml file, hence the need to add ".xml" to the end of the name.*

77 Select 'Save JSON' and name it "QuickN JSON" within the QVN folder. This JSON file is used for VisuAlign. Make sure to save the JSON file in the same folder as the brain image exports from QuPath and the Filebuilder XML file.

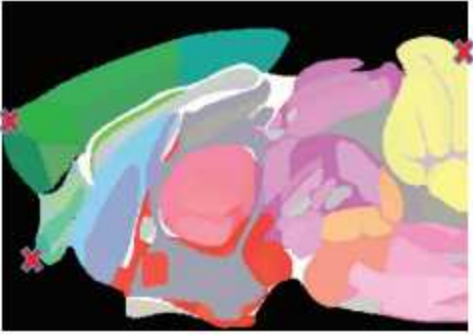
QMask

78 In the QuickNII application, go back to the first section. Show just the rainbow atlas image.

79 Adjust the horizontal plane to a hemispheric split (completely vertical). Ensure proper bisecting by confirming it within the coronal plane. You want to perfectly bisect the coronal plan. Change the angle if needed.



80 Hover the cursor over the brain viewing window and record the x-y-z coordinates (shown in the top left of the window) for the following three parts of the brain: top left, top right, and bottom left.



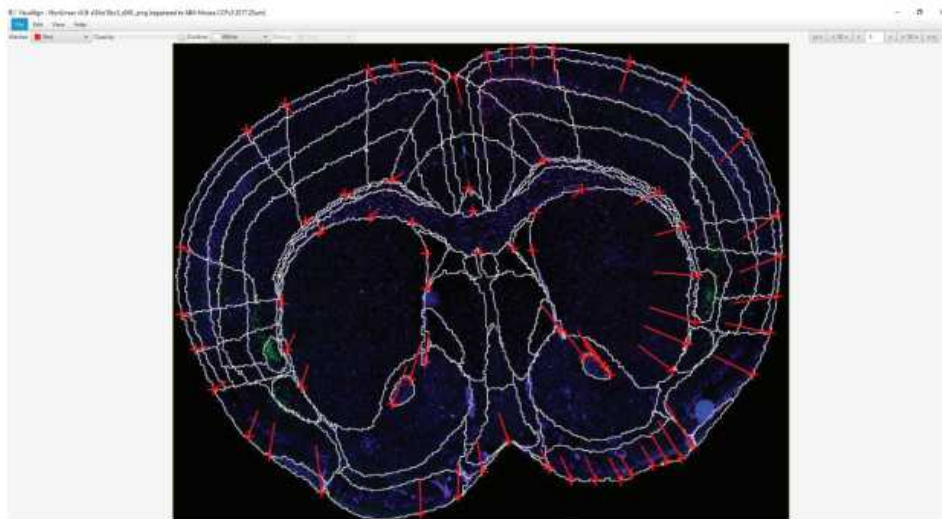
- 81 Open the QMask tool and select 'Pick XML' and open the QuickN XML file generated in QuickNII.
- 82 Enter the x-y-z coordinates.
- 83 Select 'Destination' and navigate to the Mask folder within your current project folder and select 'Open'.
- 84 Select 'Go'. The mask output should be a black and white PNG. Close QuickNII and the QMask tool.



- 85 Ensure the mask files are named using the appropriate naming convention (i.e., sl18st3sc1_s041_mask).
- 86 You can check to see if your mask outputs accurately bisect for a hemispheric split by comparing the mask output to the output from VisuAlign.

VisuAlign

- 87 Open the **VisuAlign** application.
- 88 Select 'File' > 'Open' and select the QuickN JSON file you created in QuickNII.
- 89 Drag the opacity bar all the way to the right towards 'Outline'. This will display an outline of all the regions. This is the easiest format for transformation. You can change the color of the outline and the marker for easier visualization.
- 90 Align all regions properly.
 - 90.1 Hover over an atlas region.
 - 90.2 To create a marker, click the space bar.
 - 90.3 Drag the maker symbol to the correct location.



- 90.4 Markers can be moved at any point. To delete a maker, hover over the maker and click delete on the keyboard.
- 90.5 Start with the outer alignments first then move inwards.



- 91 Select the < and > arrows at the top-right to navigate between sections. There is no need to save or store these as in QuickNII.
- 92 Once all images are complete, select 'File' > 'Save as' and save as 'VisuAlign JSON' in the QVN folder. This can be opened again to continue aligning later.
- 93 Select 'File' > 'Export' and navigate to the Atlas folder and select 'Select Folder'. This will export the files needed for **Nutil**. These are also the images that you can compare to the Mask outputs to determine the accuracy of your hemispheric split.

Nutil

- 94 Open the Nutil application and navigate to the 'Operation' tab.
- 95 Select 'New'. Select 'Quantifier' from the drop-down menu and select 'Ok'.
- 96 Select 'Save' to save the overall Nutil project (i.e., "sl24st19.nut").
- 97 Name each project using the following naming convention: 'sl#st#_classifier_left/right'.
- 98 Set the 'Segmentation folder' to the 'Input' folder within the QVN folder.
- 99 Set the 'Brain atlas map folder' to the 'Atlas' folder within the QVN folder.
- 100 Select 'Reference atlas' to the 'Allen Mouse Brain 2017'.
- 101 Set the 'XML or JSON anchoring file' to the 'VisuAlign JSON.json' file.
- 102 Set the 'Output folder' to the 'Output_left' or 'Output_right' folder within the QVN folder depending on which hemisphere you are running.

- 103 Change the 'Object colour' to match the WEB ID code for the classifier you are currently running.
- 104 Set 'Object Splitting' to 'No'.

Type	Help	Quantifier	
Name of project	Help	sl24st19_Nr4a2_left	
Analysis type	Help	QUINT	
Segmentation folder	Help	//pn.vai.org/projects_secondary/henderson/LAURA/QUINT Workflow/slide24stain19/QVN/Input	Browse...
Brain atlas map folder	Help	//pn.vai.org/projects_secondary/henderson/LAURA/QUINT Workflow/slide24stain19/QVN/Atlas	Browse...
Reference atlas	Help	Allen Mouse Brain 2017	
XML or JSON anchoring file	Help	i.org/projects_secondary/henderson/LAURA/QUINT Workflow/slide24stain19/QVN/VisuAlign_JSON.json	Browse...
Output folder	Help	//pn.vai.org/projects_secondary/henderson/LAURA/QUINT Workflow/slide24stain19/QVN/Output_Left	Browse...
Output reports	Help	All	
Object colour	Help		
Object Splitting	Help	No	
Show advanced settings >>			

- 105 In the advanced settings, set 'Custom masks (optional)' to 'Yes'.
- 106 Set the 'Custom mask folder' to the 'Mask' folder within the QVN folder.
- 107 Set the 'Custom mask colour' to either white or black depending on which hemisphere you are currently running. For output left, set the mask color to white. For output right, set the mask color to black.
- 108 Once all settings are in place, select 'Start'.

Hide advanced settings <<			
Minimum object size	Help	1	
Pixel scale (area per pixel)	Help	1	
Pixel scale unit	Help	pixels	
Custom masks (optional)	Help	Yes	
Custom mask folder	Help	//pn.vai.org/projects_secondary/henderson/LAURA/QUINT Workflow/slide24stain19/QVN/Mask	Browse...
Custom mask colour	Help		



109

Repeat for every classifier combination you have and for each hemisphere, remembering to change any settings to match. Double check that every image has a Nutil output. If outputs are missing, run Nutil again.