**BiliQML DETECTION PROTOCOL**

*Adapted from LiverQuant 1*

*Optimized on MacBook Pro (Silicone M1)(Sonoma 14.2.1)(16GB Memory)*

**I. PREPARING THE PROJECT**

1. Navigate to GitHub repository [link](https://github.com/DominickHellen/BiliQML/tree/main).
2. Download the ZIP file of contents by clicking ‘ <> Code’ 🡪 ’ ‘Download ZIP’.
3. Unpack ZIP files by right-clicking 🡪 Open.
4. All files for BiliQML should be within the ‘Code’ folder
5. Create a new folder (the directory does not matter) to perform all of your analysis
   1. Let’s call this folder ‘main’
6. Within this folder, create another new folder. This will be the hub that Qupath will use for all related paths and metadata
   1. Keep this folder entitled ‘New Folder’
7. [Download](https://qupath.github.io/) the most recent edition of Qupath and start the program
   1. Scripts tested up to Qupath v0.5.0
8. Create a project within Qupath using the 'Create Project' button
9. When prompted, navigate to the empty ‘New Folder’ created in **Step 6** and click ‘Select folder’
10. Add K19-stained whole-slide scanned images to the project using the ‘Add images’ button
11. Change ‘Set image type’ to ‘Fluorescence’, and click ‘Import’
    1. Leave the remaining image settings unchanged
12. Navigate back to, and open the ‘New Folder’ created in **Step 6** and create a folder entitled, ‘scripts’
13. Within the ‘scripts’ folder, place ‘BiliQML\_Detection.groovy’, ‘BiliQML\_Tiles.groovy’, and ‘BiliQML\_Remap.groovy’ files.
    1. These files should be available in the downloaded Github Repo folder.

**NOTE: ALL FILE NAMES MUST HAVE ONE, AND ONLY ONE, UNDERSCORE “\_” FOR THE DOWNSTREAM PYTHON CODE TO WORK PROPERLY. PLEASE ENSURE THAT THERE IS AT LEAST ONE, AND NOT MORE THAN ONE, UNDERSCORE WITHIN EACH IMMUNOFLUORESCENCE OR IHC FILE NAME**

**II. GATHERING INTENSITY FEATURES FOR DOWNSTREAM DETECTION**

1. Navigate back to Qupath, and open one fluorescence image.
2. Use the ellipse or brush annotation tool, and annotate 4-6 representative patent bile ducts within your image (**Figure 1A**).
   1. These next few steps are to gather an average intensity for each image
3. Merge all selected annotations (**Figure 1B**; left image)
4. Calculate the K19+ intensity of your annotations, as shown in **Figure 1B; middle image 🡪 right image**.
   1. Analyze 🡪 Calculate Features 🡪 Add intensity Features
   2. Click the channel associated with your K19 staining (in this case Texas Red), and then click ‘Mean’. Finally, click Run 🡪 Process all ‘Annotations’ 🡪 ‘OK’
5. The mean K19 intensity for your annotations, and thus your slide, can be found by clicking your annotation, and should be under ‘ ROI: 2µm per pixel … ‘ (**Figure 1C**)
   1. This number will be important, and utilized in the Detection portion of this protocol.
6. Repeat as necessary for additional slides – I typically do this for 2-3 slides within a project, average the intensity numbers, and use a number ~5 below the average.
   1. Example: If one image has an average intensity of 65, and another image has an average intensity of 70, I use ~63 for detection.

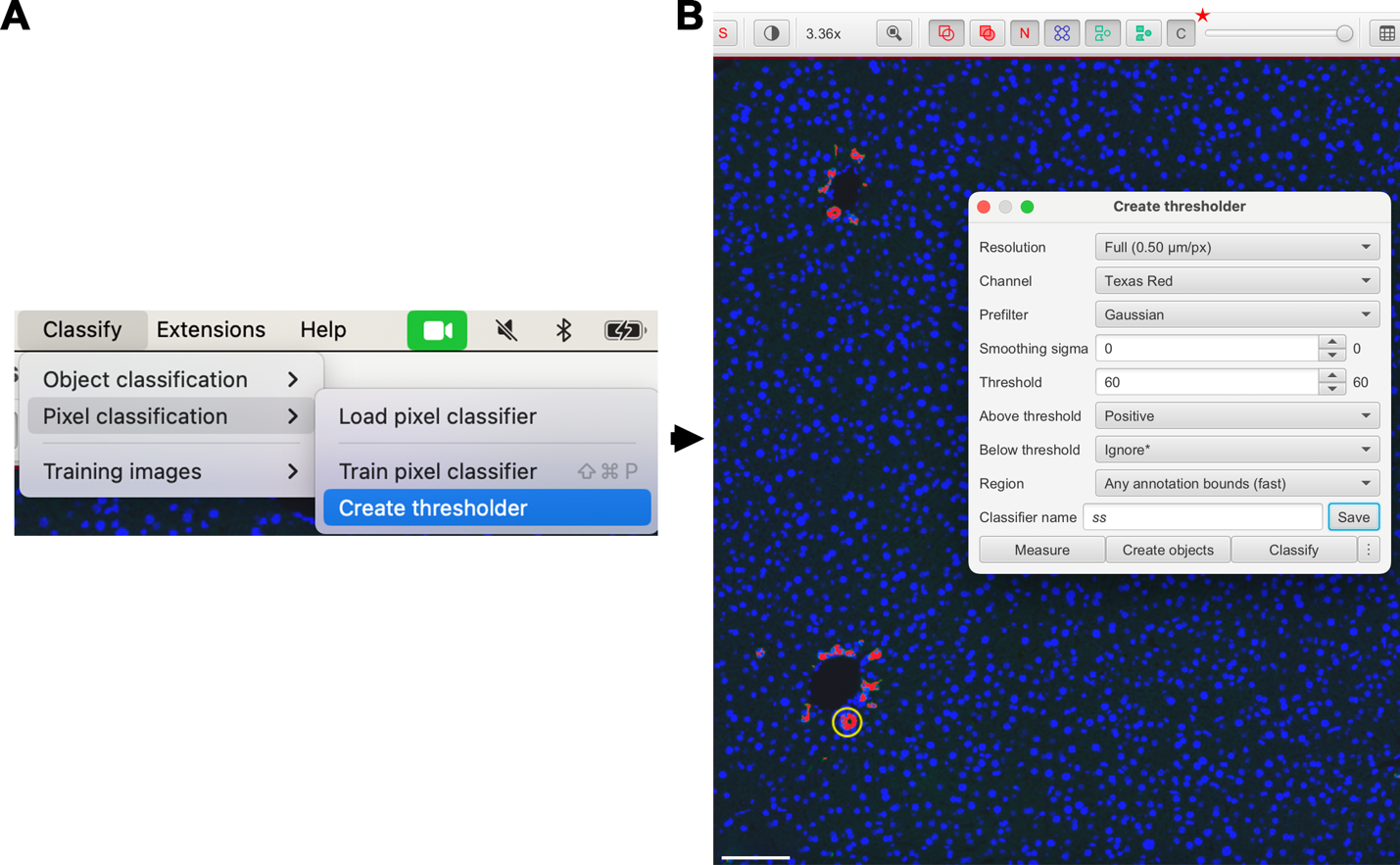
A screenshot of a computer

Description automatically generated

**Figure 1. Gathering average biliary K19 intensity. A** Representative images of bile ducts that were annotated, and used to gather an average intensity parameter. 4-6 bile duct annotations are performed on each slide, as shown in the whole-slide image to the right. Scale bars: 50µm (smaller images), 2mm (large image). **B** Merge all annotations, and calculate intensity features as shown. **C** The average K19 intensity value that will be generated from your annotations can be found by clicking ‘Annotation’, and ‘ROI: 2.00 µm per pixel…’ (STAR).

**III. CHOLANGIOCYTE DETECTION + SLIDE ANNOTATION**

1. Classify K19+ staining by going to Classify 🡪 Pixel Classification 🡪 Create Thresholder (**Figure 2A**)
   1. Turn on- and off- the positive pixel overlay if necessary using the ‘c’ button in the tools tab above the image.
2. A pop-up will show up, ensure all settings are *exactly* the same as those shown in **Figure 2B**. See below for where it would make sense to change the settings.
   1. Change Texas Red to whatever channel you stained cholangiocytes in – in this case, it was Texas Red.
   2. Threshold should be adjusted to whatever number was generated in the ‘Gathering Intensity Features’ step – in this case, it was 65, but I lowered that number to 60 (lowering the threshold number by ~5 helps pick up Type 1 forms that don’t always robustly stain positive for K19)
3. Save this tresholder as ‘ss’ (Figure 2B). DO NOT MEASURE, or CLASSIFY anything from here.



**Figure 2. Creating a K19+ pixel classifier. A** Open up a thresholder in Qupath, as shown. B Create a thresholder using the same settings as detailed. Ensure that the classifier is saved as ‘ss’. To hide, or show, the accuracy of the threshold that you are applying, click off, and on, the ‘c’ button (star). Scale bar: 100µm.

1. After you have saved your K19+ classifier, delete all annotations.
2. Proceed to annotate all the slides in your dataset using the appropriate annotation tools.
   1. Do not just annotate bile ducts, annotate the entire whole-slide image, or wherever you are interested in quantifying the biliary landscape
   2. Ensure that you save changes for each slide after you are finished annotating
3. When you are finished annotating all your slides, go to a random image that you have already annotated.
4. Click Automate 🡪 Project Scripts 🡪 BiliQML\_Detection (**Figure 3A**)
   1. This should open up the script editor, with the BiliQML Detection
5. Right click 🡪 Run 🡪 Run for Project (**Figure 3B**)
   1. Select all the images that you annotated for the project, and move them from the Available window 🡪 Selected Window, then hit ‘OK’
6. Importantly, wait for the script to finalize running its analysis. If you try to mess with the program while it is running this script, it will crash.
   1. It is best to just run the script and leave your laptop alone while it runs.
7. When all cholangiocytes are detected, it is time to export these images. Click Automate 🡪 Project Scripts 🡪 BiliQML Tiles (**Figure 3C**)
8. Right click 🡪 Run 🡪 Run for Project (**Figure 3D**)
   1. After this script runs, all the tiles for your project should be in a folder entitled ‘tiles’ in your Qupath Folder.
   2. If your images are smaller (less than 5120 pixels), you will receive an error, or nothing will export. Scroll down to .tileSize(5120) and change 5120 to 2560, 1280, 640, 320, etc., until the code begins to export tiles. Use the utmost number of pixels (2560, 1280, 640, 320, etc.,) that leads to the successful export of your tiles.
9. Important last step here – In Qupath, go to Measure 🡪 Export measurements, and select all the files that you have done analysis on, and move them over to ‘selected’. Before exporting measurements, change export type to ‘Annotations’ and change separator to ‘Comma (.csv)’. Finally, click Export.
   1. You will need this file to get number of BiliQML detections per mm2 in **Section VI**.
   2. The output file destination can be anywhere except the ‘tiles’ folder.

A screenshot of a computer

Description automatically generated

**Figure 3. Creating and exporting biliary detection tiles. A** Open up the BiliQML detection script, and **B** Run for project. When this is completed, **C** Open up the BiliQML tiles script, and **D** Run this script for your entire project. All images should have a specific folder, that contains segmented detections and tiles, within the tiles folder that is created from running this script.

**IV. BiliQML PROCESSING**

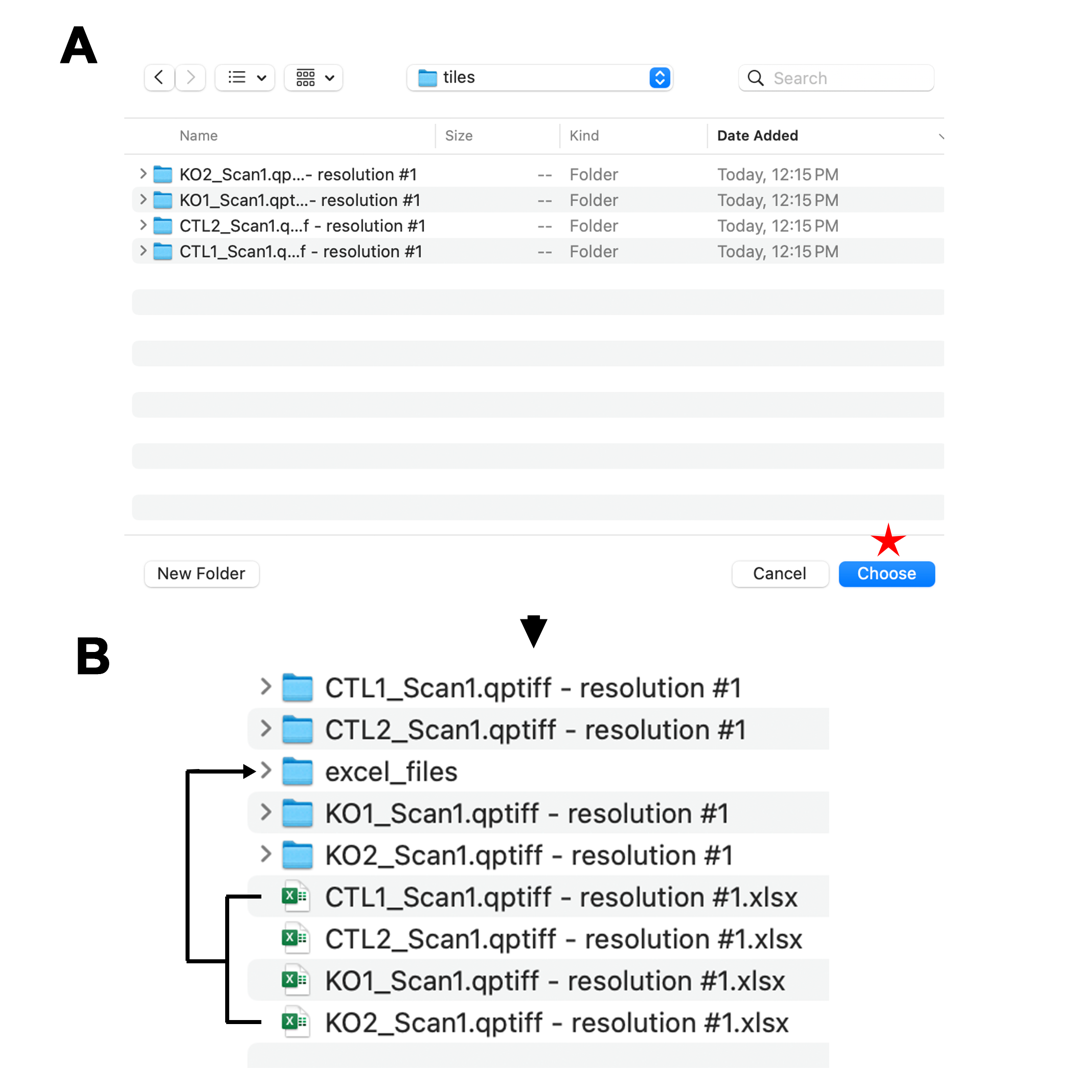
1. Download the latest version of [Python](https://www.python.org/downloads/).
2. Download the latest version of [PyCharm](https://www.jetbrains.com/pycharm/download/?section=mac).
3. Open PyCharm and click New Project
   1. Ensure that the project is using a ‘virtualenv’ and that the base interpreter is Python 3.9.
4. Go to File 🡪 Open… and open the “BiliQML\_Processing.Py” script.
   1. If using IHC, use “BiliQML\_Processing\_IHC.Py”
5. Go to the Terminal window within your Python Project (**Figure 4A**).
6. There is a list of packages that the user must install for this script to work properly. To find each package that must be installed, go to the BiliQML\_Packages.rtf file and open it.
7. Copy and paste each pip command *separately* from the .rtf file into the terminal window (**Figure 4B**).

A screenshot of a computer

Description automatically generated

**Figure 4. Preparing the BilliQML Processing script. A** Open the terminal window in your Python environment, and **B** paste in each Pip install commad.

1. When everything is installed, the script should run normally. Click the Run button ().
2. This will open up a pop-up. Find the ‘tiles’ folder that was generated in **III.11**, and open it so that all the folders associated with your images are viewable (**Figure 5A**).
3. Click ‘Choose’
4. After this, the program will take anywhere from ~3-10minutes per folder, and create an excel file that labels each detection within each ‘folder’ (whole slide image) with appropriate metrics for downstream decision-making (**Figure 5B**).
   1. *This is the most difficult and CPU-intensive step (also high possibility for error), so if you are having trouble, please do not hesitate to contact me:* [*dhellen@mit.edu*](mailto:dhellen@mit.edu) *and we can zoom and figure it out.*
5. Move all of the excel files into the ‘excel\_files’ folder.



**Figure 5. BiliQML processing results. A** When presented with the pop-up generated from running the BiliQML\_processing script, navigate to the tiles folder with all of your image folders, and click ‘Choose’ (star). **B** When the BiliQML\_Processing script is finished running, all of the excel files, needed for the BiliQML\_Quant script will be in this folder.

**V. BiliQML QUANTIFICATION**

1. Install [Anaconda Navigator](https://www.anaconda.com/download) and open it
2. Install and launch Jupyter Notebook
3. Navigate to the BiliQML Code folder that you have downloaded, and open ‘BiliQML\_Quant.ipynb’
4. Each cell of code is required to properly run this script properly, so start with cell 1.
   1. There will be packages that you need to install to ensure cell 1 loads properly.
5. At this point, identify where you installed the BiliQML package contents. Navigate there, and find the directory path associated with ‘Data’. In this folder, there is an excel file that has all the ***labeled*** training data: ‘TRAIN7.xlsx’. There is also a folder that contains the ***unlabeled*** training data with metadata: ‘TRAINall’.
6. In cell 2, place the directory path associated with TRAIN7.xlsx on line 4, and the directory path associated with TRAINall on line 7.
7. Proceed with each following cell, until you reach cell 17.
8. In cell 17, place the path directory of the folder where your excel files are. This folder was created in **IV.14**.
9. Run cell 18, skip cell 19 (this cell is for spatial purposes).
   1. This cell uses the trained Gradient\_Boosted classifier 2, 3 to label each detection as a biliary form.
   2. Spatial mapping will be covered in **VII**.
10. In cell 20, name the file that will contain all of the BiliQML detections (line 4), then run this cell.
    1. The excel file with all of your detections will be in the Data folder that was used in **V.5**.

**VI. BiliQML STATISTICAL ANALYSIS**

***cb = Type 3, dp = Type 2, sc = Type 1, m = nonclass***

1. Open the excel file with all the detections, generated in **V.10**.
2. Open the excel file with all the measurements, generated in **III.12**.
3. Within the excel file from step 1, navigate to sheet 1, entitled “Counts, additional scans”
4. Sort column 0 alphabetically, and expand the selection.
5. Now, navigate to the excel file with all the measurements, and copy the file name, and ‘Area µm2’ for each file.
6. Paste this data in a column next to the counts data (**Figure 6A**).
7. Now, copy cb data into the column adjascent to the Area µm2 column. Repeat this for dp, sc, and m (Figure 6B).
8. To normalize each value, divide the number of each subgroup by the µm2 data, and multiply by 1 million.
   1. Multiplying by 1 million will give how many datapoints there are per mm2, rather than µm2.
9. Total biliary forms is calculated by adding cb, dp, and sc from each sample. Normalization follows by dividing by the µm2 data and multiplying by 1 million, similar to step 8.
10. Data points are taken from the highlighted columns in **Figure 6B** into prism and compared in Prism.
11. The percent of each class (cb,dp,sc,m) in each sample is found in sheet 2, entitled “Percents, additional scans”
12. To get mean area for each class, go to sheet 3, entitled “Mean Area per Class (All Scans)” and sort column “Class Assigned” alphabetically.
13. Next, multiply the “Area” column by 0.225, as seen in **Figure 6C**. This normalizes the Area metric.

A screenshot of a document

Description automatically generated

**Figure 6. Representative preparation for statistical comparisons.** **A** Sort detections of each sample alphabetically, and set up your excel sheet as shown. **B** Normalize the number of detections per sample by dividing by the Area µm2 value, and multiplying by 1,000,000. **C** Normalize the Area of each class for each sample by multiplying by 0.225.

**VII. SPATIAL RE-MAPPING OF BiliQML DETECTIONS**

*Can only map the detections of one slide at a time, cannot do this in bulk*

*This process deletes the original annotation, so ensure that you have that data saved somewhere*

***cb = Type 3, dp = Type 2, sc = Type 1, m = nonclass***

1. Choose a sample where you would like to map back BiliQML detections onto the original slide. Create a new folder, wherever is best, and place the excel\_file (these excel files should be available in the folder created in **III.12**) associated with the sample inside this folder.
2. Identify the path name associated with the excel file for your sample, paste it into cell 17, and run this cell.
3. Now, run cell 18, and then run cell 19.
   1. Cell 19 adds centroid data and coordinates together and attaches them to each detection.
4. After this is completed, create an empty folder. This folder will be used by Qupath to map back each detection.
5. Type the directory of that empty folder, followed by the EXACT name of the file you are interested in within line 14 of cell 21. After this, run this cell.
   1. The file name should match exactly with the file name in Qupath that you want to map back detections to, followed by ‘.csv’ at the end
6. Now, Navigate back to Qupath, and open the file you are interested in.
7. Create new Annotation labels by going to Annotations 🡪 … 🡪 Add/Remove… 🡪 Add Class (**Figure 7A**). Add classes cb, dp, sc, m.
8. Open the BiliQML\_Remap.groovy script within Qupath. This should be under Automate 🡪 Project Scripts…
9. Go to line 34 within this script, and paste in the exact path, with file name, to the file generated in step 5.
10. Run this script, this should take ~3-6 minutes.
11. To visualize, click Analyze 🡪 Density maps 🡪 Create Density Maps
12. To visualize specifically bile ducts, set object type to ‘All detections’ and Main class to ‘cb’ (**Figure 7B**).
    1. If you want to compare control versus treated samples using this method, ensure that the ranges are the same across comparisons (**Figure 7B**).

A screenshot of a computer

Description automatically generated

**Figure 7. Settings for spatial remapping of BiliQML detections.** A Add appropriate cb, dp, sc, and m annotation classes prior to running the BiliQML\_Remap script. **B** After running the script, to visualize, via density maps, separate classes of BiliQML detections, change the ‘Main class’ setting (green star). When comparing control and treated samples, ensure the ranges utilized across each sample are the same (red star). Scale bar: 2mm.

**PLEASE DO NOT HESITATE TO EMAIL ME WITH ANY QUESTIONS:** [**DHELLEN@MIT.EDU**](mailto:DHELLEN@MIT.EDU)

**REFERENCES**

1. Hellen DJ, Karpen SJ. LiverQuant: An Improved Method for Quantitative Analysis of Liver Pathology. Bio-protocol. 2023;13(14).

2. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V. Scikit-learn: Machine learning in Python. the Journal of machine Learning research. 2011;12:2825-30.

3. Van der Walt S, Schönberger JL, Nunez-Iglesias J, Boulogne F, Warner JD, Yager N, Gouillart E, Yu T. scikit-image: image processing in Python. PeerJ. 2014;2:e453.