

User Guide

Digital Resurrection of Historical Maps Using Artificial Intelligence Project **Choropleth Map Segmentation Tool**

Designed for the Pacific Forestry Centre, Natural Resources Canada
Authored by Team FourTrees, Camosun College ICS Capstone 2020

Version 0.02 (August 2020)

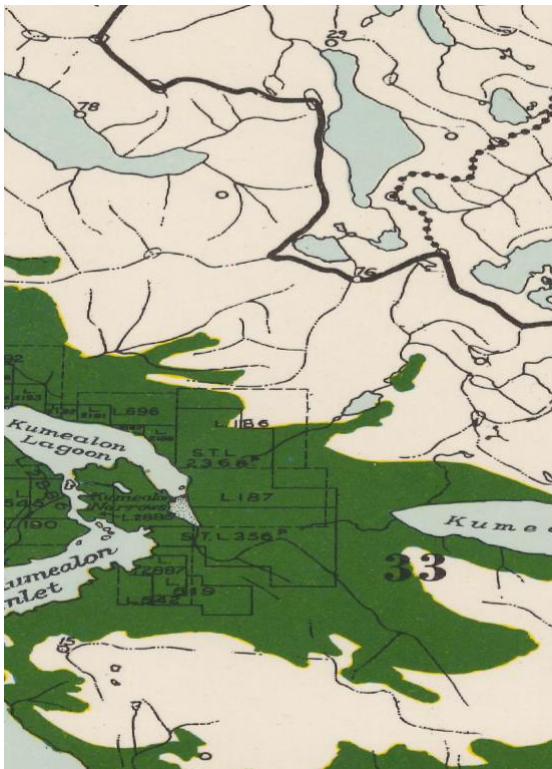


Table of Contents

Introduction	1
CMS Tool Overview	1
Instructions.....	2
Installation & Setup	3
File Management	6
Warning.....	6
Preparing Map Files for Input	7
A. Direct Download.....	7
B. Local or Network Copying.....	7
Using the CMS Tool.....	8
A. Single Image Processing Option:	9
B. Batch Image Processing Option:	12
Example of Segmentation Results.....	14
Viewing Shapefiles.....	15
Summary	23
Conclusion	24
Appendix	25
K-Means vs HSV Segmentation Methods	25

List of Figures and Tables

Figure 1 – Install Anaconda for your operating system.	3
Figure 2 – Confirm Anaconda installation.	3
Figure 3 – Download the CMS Tool git repository.	4
Figure 4 – Folder structure of the cloned CMS Tool git repository.	4
Figure 5 – Install the conda environment.	4
Figure 6 – Confirm creation of drhmai environment.	4
Figure 7 – Activate the conda environment.	5
Figure 8 – Download a map geoTIFF using the command-line interface.	7
Figure 9 – CMS Tool Process Flowchart.	8
Figure 10 – Select the number of input images to process.	9
Figure 11 – Choosing to crop an image.	9
Figure 12 – Choosing a segmentation method.	10
Figure 13 – Completing the HSV segmentation and file conversion processes.	10
Figure 14 – Choosing a threshold value.	10
Figure 15 – Observing the threshold value on a pixel intensity histogram plot.	11
Figure 16 – Choosing a ‘k’ value for the number of clusters.	11
Figure 17 – Completing the k-means segmentation and file conversion processes.	11
Figure 18 – Selecting the batch processing option.	12
Figure 19 – Example of processing output files after image processing is complete.	12
Figure 20 – Example of segment output files after image processing is complete.	13
Figure 21 – Example of shapefile output files after image processing is complete.	13
Figure 22 – Example input image 092O_NESE_P_35_cropped.tif cropped to size 2000 x 2000 pixels.	14
Figure 23 – Example output after HSV segmentation.	14
Figure 24 – Example output after k-means segmentation with a threshold of 80 and k-value of 5.	14
Figure 25 – Create a new project in QGIS.	15
Figure 26 – Add a vector layer.	15
Figure 27 – Select a shapefile to view.	16
Figure 28 – Map shapefile after being loaded into QGIS.	16
Figure 29 – Navigate to the vector layer properties.	17
Figure 30 – Navigate to the “Load Style” menu.	17
Figure 31 – Load style from file.	18
Figure 32 – Symbology tab after loading style file.	18
Figure 33 – Map shapefile with style applied.	19
Figure 34 – Add a raster layer.	19
Figure 35 – Select a raster data source.	20
Figure 36 – Navigate to the raster layer properties.	20
Figure 37 – Modify raster opacity.	21
Figure 38 – Map raster and shapefile layers blended together.	21
Figure 39 – Detailed map section with raster and vector layers blended together.	22
Figure 40 – Detailed map section of the raster layer.	22
Figure 41 - Detailed map section of the vector layer.	22
Figure 43 – CMS Tool Workflow.	23
Table 1 – Comparison Summary between K-Means Clustering and HSV Segmentation Methods.	25

Introduction

Welcome to the User Guide for the Choropleth Map Segmentation Tool.

The Choropleth Map Segmentation (CMS) Tool is a command-line interface tool that is designed to convert a specific digitized set of hand-drawn, historical choropleth maps into a modern, analysis-ready format.

This guide will provide you with introductory information and walk you through the necessary steps to install and run the tool on your machine.

- To learn more about the CMS Tool, refer to the **CMS Tool Overview**.
- To install the tool, refer to

- **Installation & Setup.**
- To run the tool, refer to

- **Using the CMS Tool.**

Details regarding tool development and source code are available in the **Appendix** section at the end of this document and may be of interest to advanced users. Please note that it is not necessary to review or understand this material in order to use the CMS Tool.

CMS Tool Overview

The Choropleth Map Segmentation (CMS) Tool is designed to convert a specific digitized set of hand-drawn, historical choropleth maps into a modern, analysis-ready format. These maps date back to the 1950s and contain data on historical forest cover in British Columbia. They currently exist as scanned geo-referenced Tagged Image File Format (geoTIFF) files, however, this format has limited potential for research and analysis. Converting the map geoTIFFs to a geospatial vector data format – such as a shapefile – can help resolve these limitations, as shapefiles offer greater flexibility for data analysis and storage.

The CMS Tool reads in map geoTIFF images and analyzes the colour data across all pixels. It applies image segmentation methods to identify forest cover categories by inspecting and labelling each pixel with the appropriate category, based on the original legend for the historical map set. Pixels that cannot be classified into these categories are grouped and removed, while categorized pixels, or segments, are de-noised to remove any misclassified or outlier pixels. Finally, the finished colour segments are converted and exported separately as shapefiles.

The CMS tool is a product of the Digital Resurrection of Historical Maps Using Artificial Intelligence (DRHMAI) project. This project is sponsored by the National Forest Information System (NFIS) team at the Pacific Forestry Centre for Natural Resources Canada, and is part of the Camosun College Information & Computer Systems (ICS) Capstone 2020 showcase.

Instructions

The instructions documented in this section cover the installation and operation of the CMS Tool on Unix-based operating systems.

It is assumed that you have access to a computer, a working internet connection, basic familiarity with using the command-line interface, and access permissions to perform the following actions:

- Download files from Github.
- Install open-source software and associated libraries.
- Create directories and files.
- Run commands using the command-line interface.

This section is organized sequentially with instructions grouped by topic:

- To view instructions on retrieving the CMS Tool source code and installing Anaconda and Python, refer to

- **Installation & Setup.**
- To learn about the file structure of the CMS Tool, refer to

- **File** Management.
- For instructions on gathering and preparing geoTIFF images for processing, refer to

- **Preparing Map Files for Input.**
- For instructions on running the CMS Tool using command-line, refer to

- **Using the CMS Tool.**
- To view and analyze the CMS Tool output, refer to **Viewing Shapefiles.**
- For a quick overview of the entire instruction process, refer to the

Viewing Shapefiles

Shapefile outputs from the CMS Tool can be viewed using GIS software such as QGIS.

- Step 1. Install QGIS. This user manual assumes that version 3.10 is installed.
- Step 2. Launch the application.
- Step 3. Select “Project->New” from the application menu as shown in Figure 25.

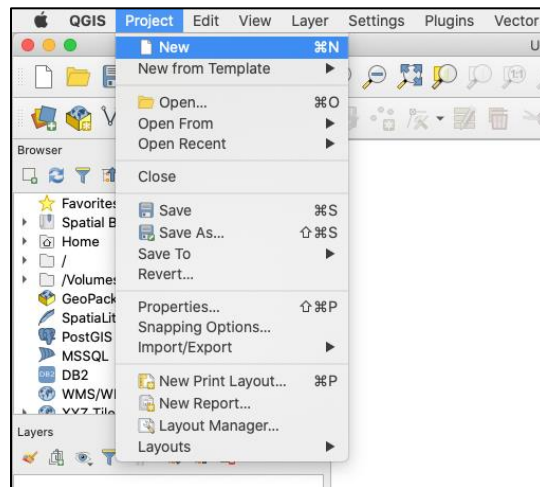


Figure 25 – Create a new project in QGIS.

- Step 4. Select “Layer->Add Layer->Add Vector Layer...” as shown in Figure 26.

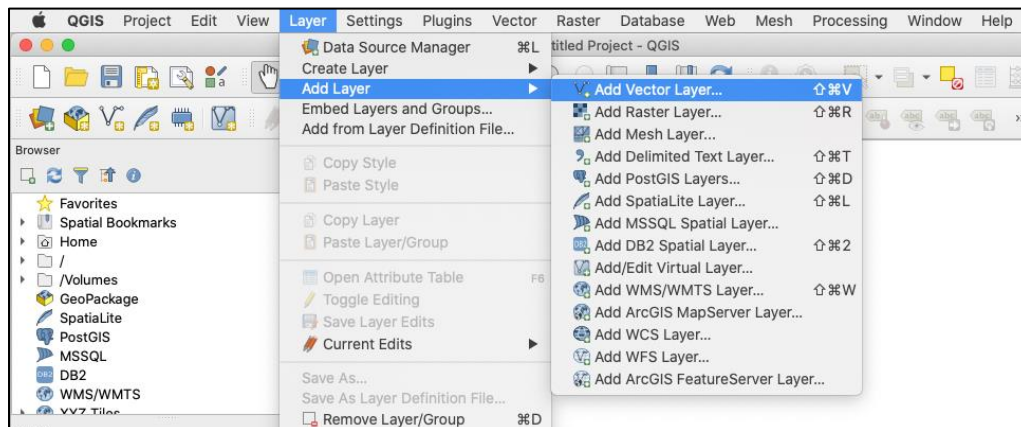


Figure 26 – Add a vector layer.

- Step 5. Select the map's shapefile from the CMS Tool's output directory in `~/choropleth-map-segmentation/output/<map>/shapefiles/<map>.shp`. Be sure to substitute `<map>` with a specific map of interest, e.g. `0920_NESE_P_35_cropped` as shown in Figure 27.

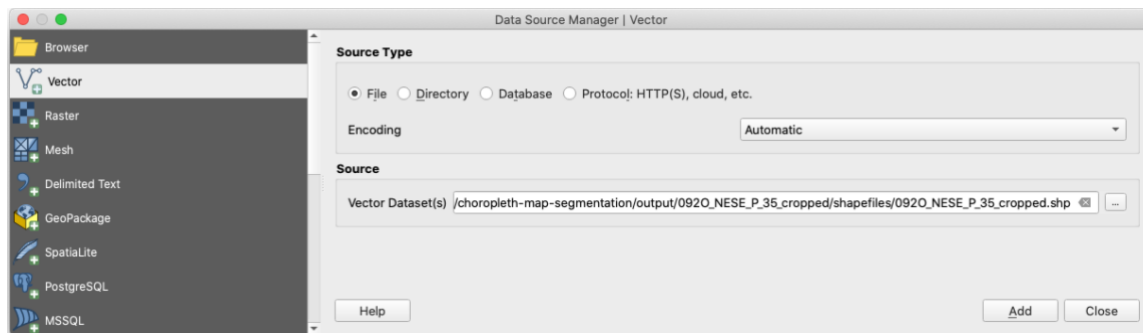


Figure 27 – Select a shapefile to view.

- Step 6. Click the “Add” button. The shapefile will appear on the screen as shown in Figure 28.

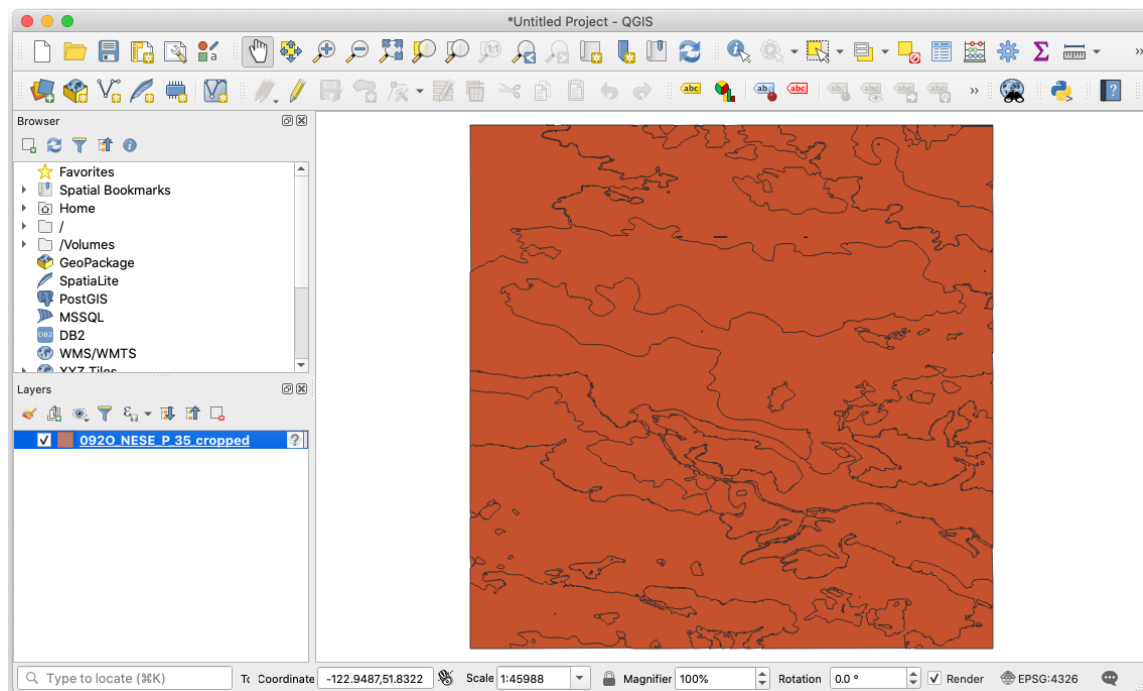


Figure 28 – Map shapefile after being loaded into QGIS.

- Step 7. Right-click the new layer from the Layers panel and select “Properties...” as shown in Figure 29.

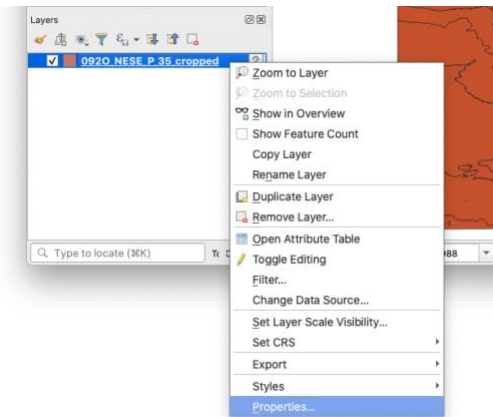


Figure 29 – Navigate to the vector layer properties.

- Step 8. Click “Style->Load Style...” from the bottom of the “Symbology” tab as shown in Figure 30.

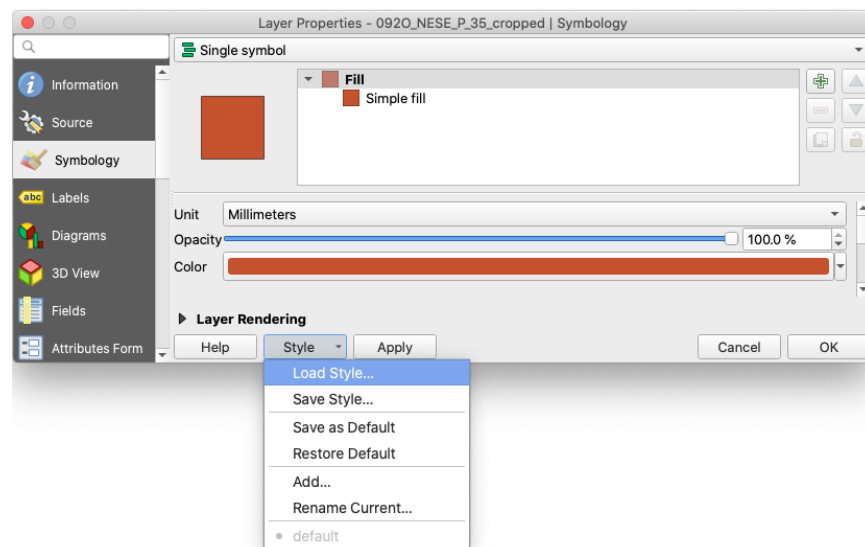


Figure 30 – Navigate to the “Load Style” menu.

- Step 9. Select the style `qgis_style.qml` file from the `~/choropleth-map-segmentation/config` folder as shown in Figure 31.

Note – This style is configured to work with HSV-segmented shapefiles only. For shapefiles generated using k-means segmentation a new custom style will need to be created (which is beyond the scope of this user guide).

Step 10. Click “Load Style”. The style will be loaded and the Style Manager will automatically close.

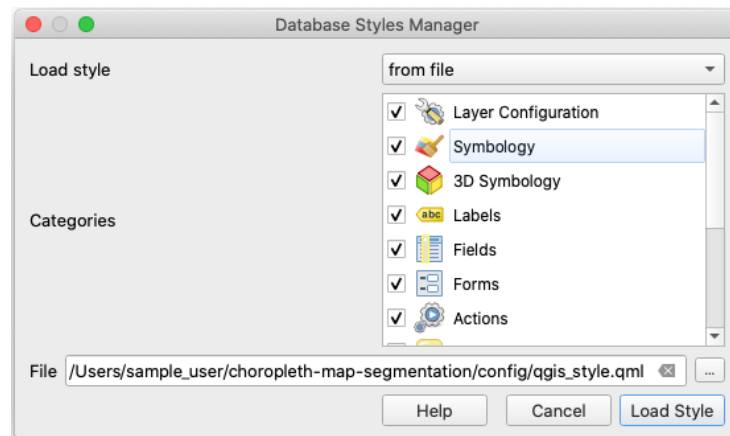


Figure 31 – Load style from file.

Step 11. The “Symbology” tab of the Layer Properties menu will now appear as shown in Figure 32.

Step 12. Click “OK” to close the Layer Properties window.

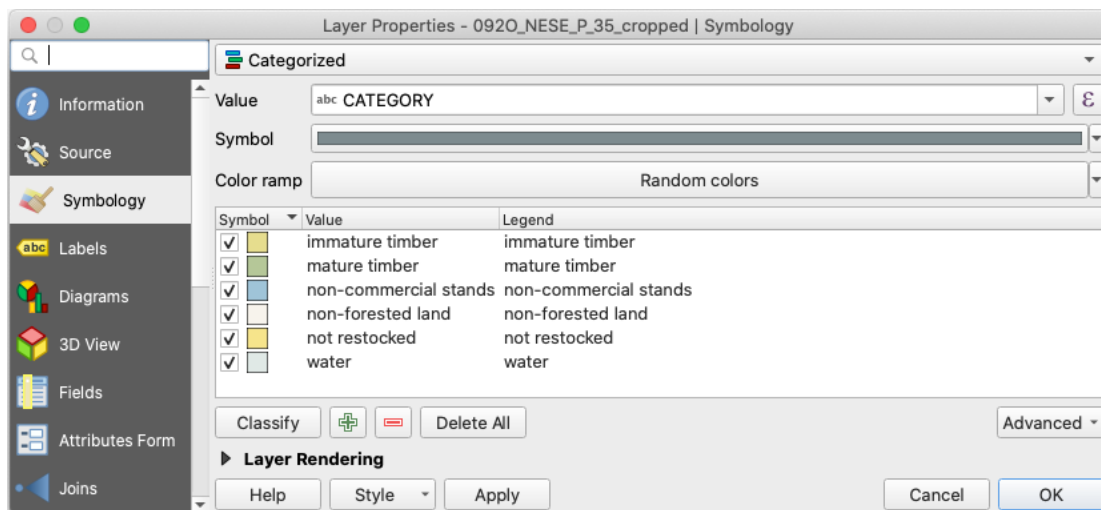


Figure 32 – Symbology tab after loading style file.

The map will appear as shown in Figure 33 with the different forest cover regions coloured and labelled.

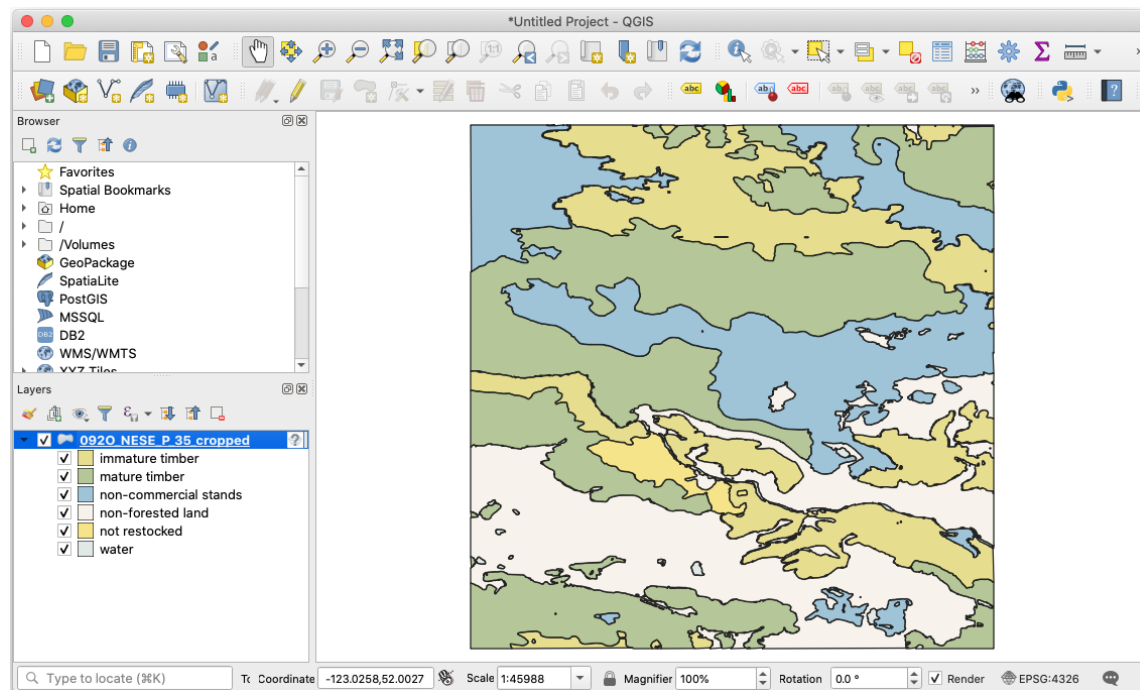


Figure 33 – Map shapefile with style applied.

The shapefile has been successfully loaded and styled and is ready for analysis. To compare the shapefile to the original raster map, proceed with steps ## to ##

Step 13. From the menu, select “Layer->Add Layer->Add Raster Layer...” as shown in Figure 34.

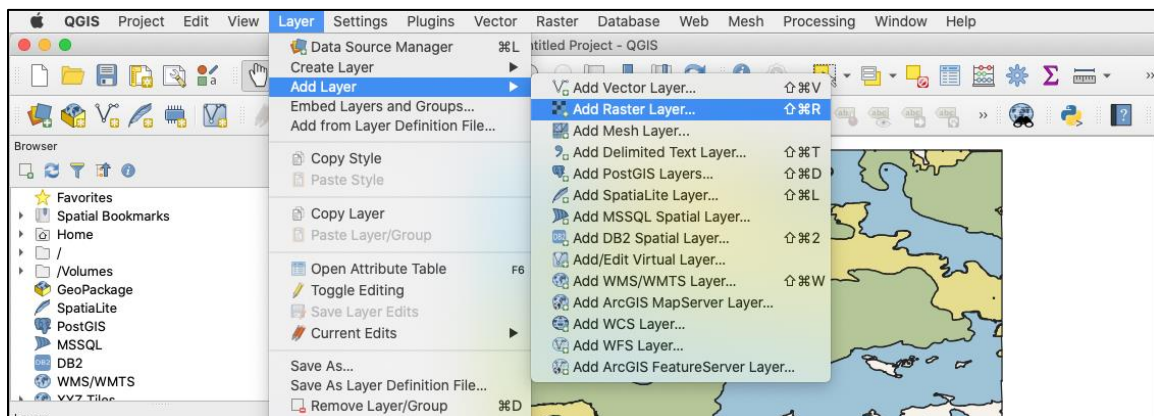


Figure 34 – Add a raster layer.

Step 14. In the “Source type” section, select “Protocol: HTTP(S), cloud, etc.” and enter the map URI, e.g. `opendata.nfis.org/bc/interim_forest_cover/0920_NESE_P_35_cropped.gtiff` as shown in Figure 35.

Alternatively, select the “File” option and select a local file from the “choropleth-map-segmentation/input” directory or elsewhere.

Step 15. Click Add.

Step 16. Click Close. The map should appear in the foreground as a new layer.

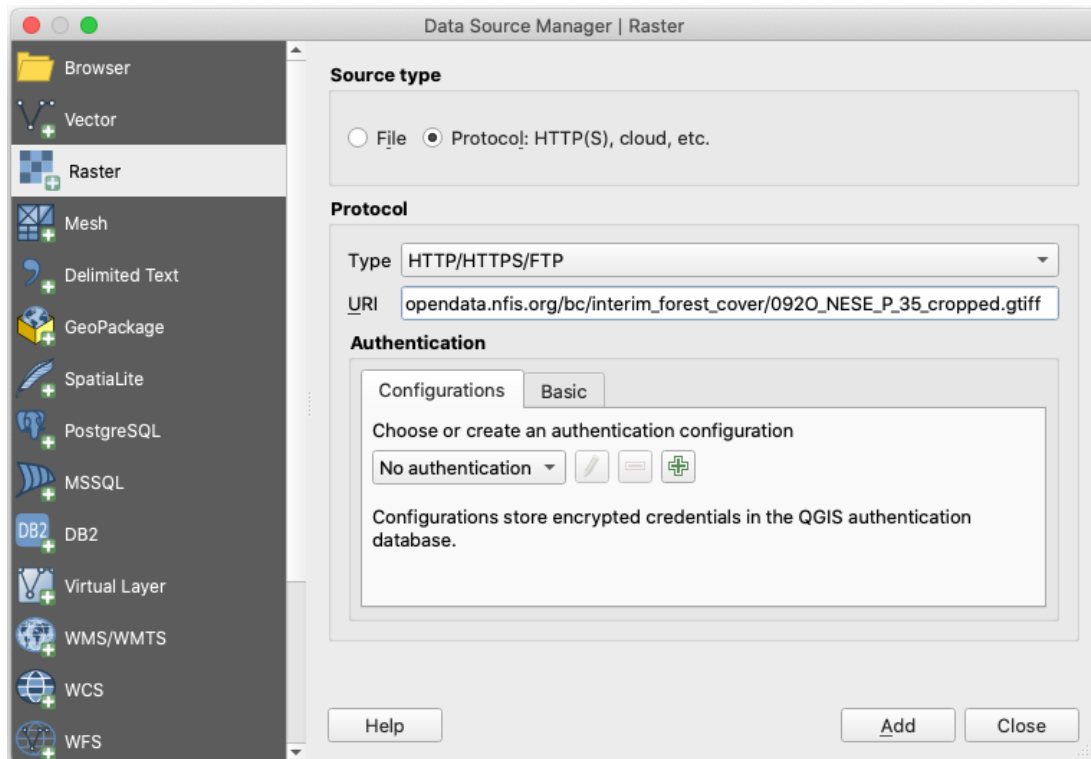


Figure 35 – Select a raster data source.

Step 17. Right-click the new layer from the “Layers” panel and select “Properties...” as shown in Figure 36.

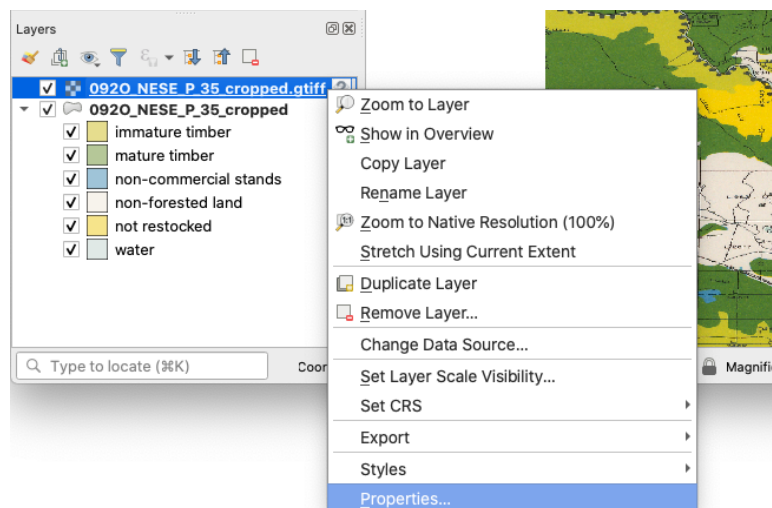


Figure 36 – Navigate to the raster layer properties.

Step 18. In the “Transparency” tab, set the Global Opacity to 50% as shown in Figure 37

Step 19. Click OK.

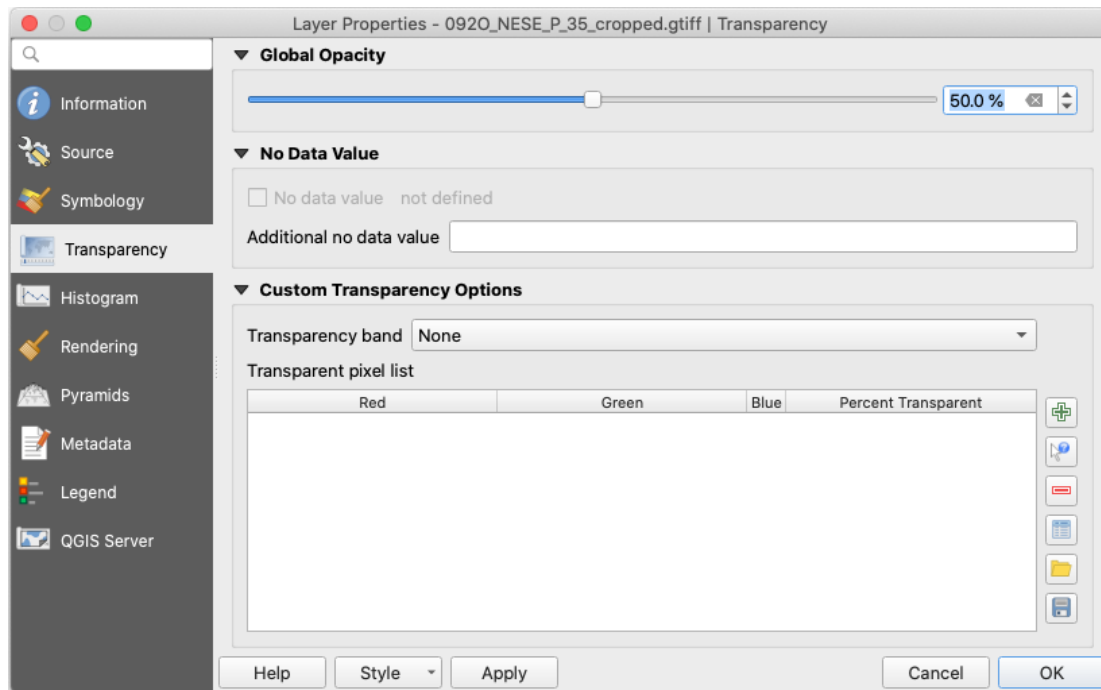


Figure 37 – Modify raster opacity.

The raster image’s colours should now appear to be lighter in colour and blended with the shapefile as shown in Figure 38.

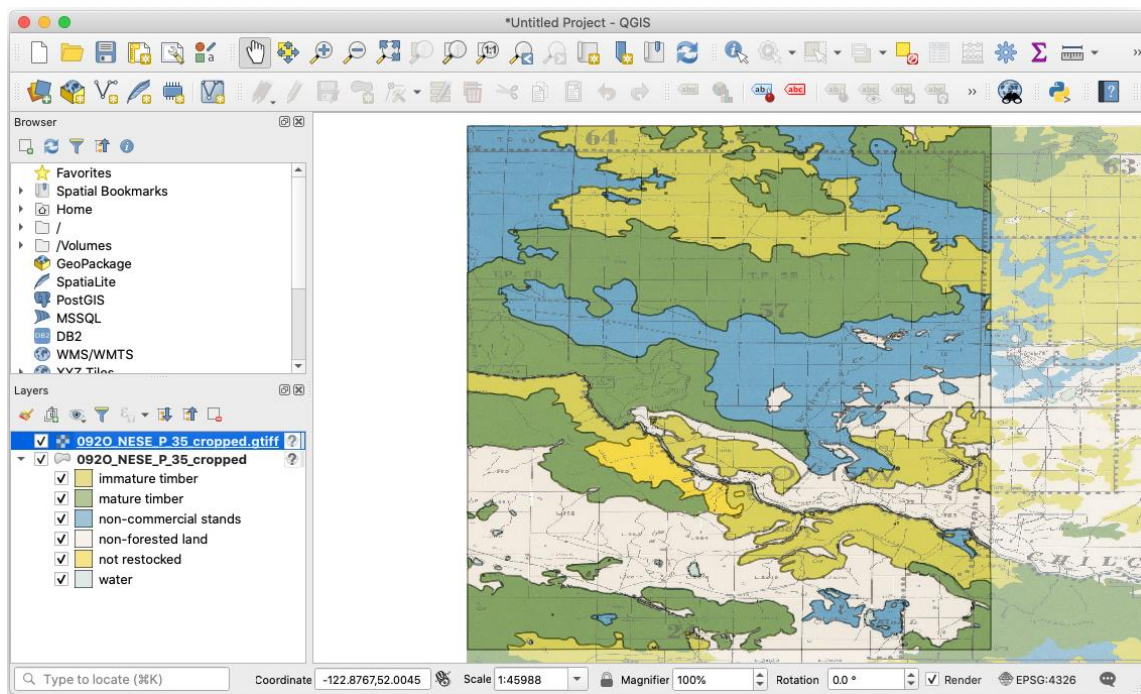


Figure 38 – Map raster and shapefile layers blended together.

- Step 20. Zoom in to a small sub-section of the map, e.g. the section shown in Figure 39.
- Step 21. Click the check-mark to the left of the vector layer in the Layers panel. This will hide it from the view. The map will now appear similar to Figure 40.
- Step 22. Click the check mark to the left of the vector layer again to unhide it.
- Step 23. Click the check-mark to the left of the raster layer in the Layers panel. This will hide it from the view. The map will now appear similar to Figure 41.

By examining the blended view (Figure 39) as well as hiding the and unhiding the raster and vector layers separately (Figure 40, Figure 41) allows the user to make a detailed comparison between the original raster image and the shapefile output from the CMS Tool.

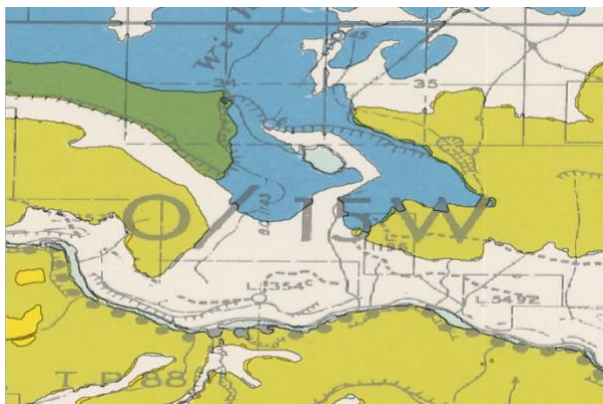


Figure 39 – Detailed map section with raster and vector layers blended together.

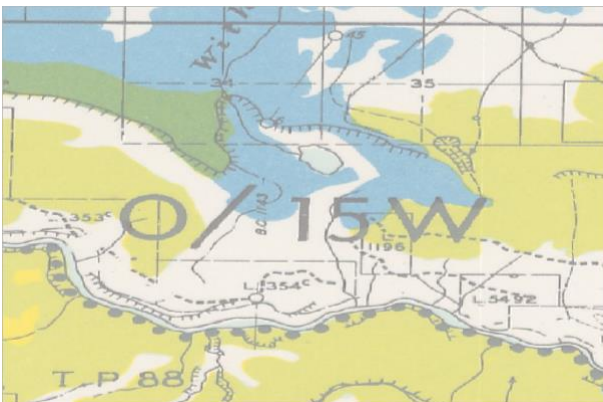


Figure 40 – Detailed map section of the raster layer.

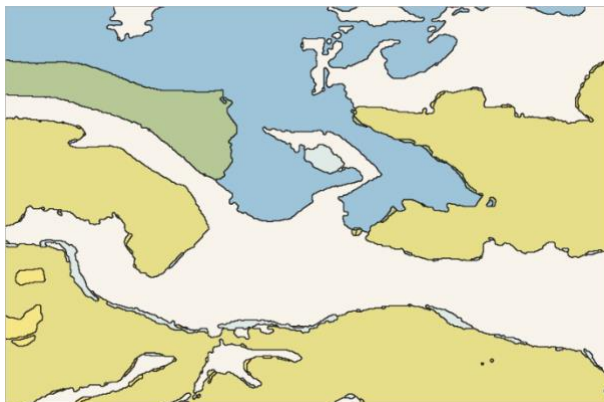


Figure 41 – Detailed map section of the vector layer.

- Summary.

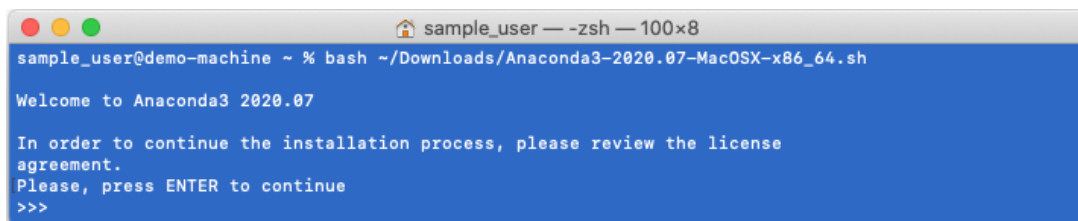
Installation & Setup

The CMS Tool runs inside an Anaconda environment that contains all the Python libraries it requires to operate. This pre-configured environment is provided with the CMS Tool source code in the file `drhmai_environment.yml` so that it can easily be duplicated.

Please note that installing Anaconda may take a few minutes and about 3.5 GB of storage space.

- Step 1. Download the latest version of Anaconda for your operating system from the [Anaconda website](#)¹.
- Step 2. Install Anaconda by following the instructions provided in the [Anaconda documentation](#)². An example of the command-line installation for macOS is shown in Figure 1.

Note – Python 3 is installed with Anaconda and does not have to be set up separately.



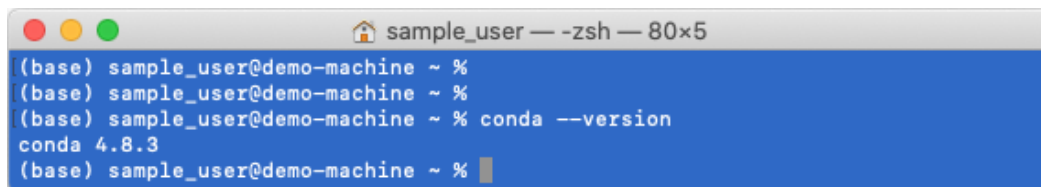
```
sample_user@demo-machine ~ % bash ~/Downloads/Anaconda3-2020.07-MacOSX-x86_64.sh

Welcome to Anaconda3 2020.07

In order to continue the installation process, please review the license
agreement.
Please, press ENTER to continue
>>>
```

Figure 1 – Install Anaconda for your operating system.

- Step 3. Restart the command-line terminal application.
- Step 4. Verify that the installation was successful by entering the command `conda --version` as shown in Figure 2.



```
(base) sample_user@demo-machine ~ %
(base) sample_user@demo-machine ~ %
(base) sample_user@demo-machine ~ % conda --version
conda 4.8.3
(base) sample_user@demo-machine ~ %
```

Figure 2 – Confirm Anaconda installation.

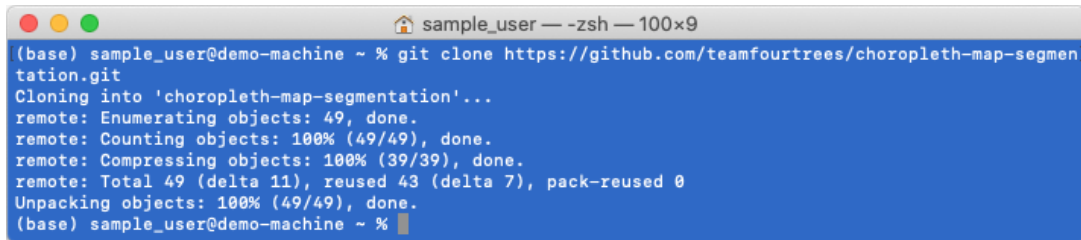
- Step 5. From the command-line, clone the CMS tool git repository by running the command `git clone https://github.com/teamfourtrees/choropleth-map-segmentation.git` as shown in Figure 3.

Note – If you are unable to use the git command, check that you have [Git installed](#)³.

¹ Anaconda download page: <https://www.anaconda.com/products/individual>

² Anaconda installation guide: <https://docs.anaconda.com/anaconda/install/>

³ Git installation guide: <https://git-scm.com/book/en/v2/Getting-Started-Installing-Git>



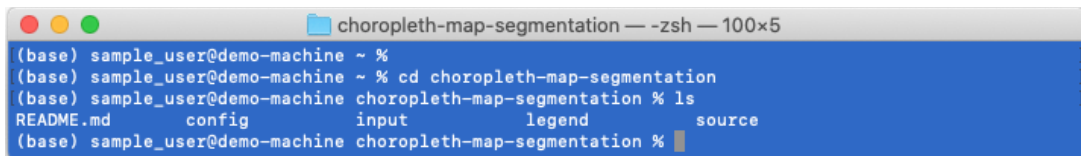
```

sample_user — -zsh — 100x9
(base) sample_user@demo-machine ~ % git clone https://github.com/teamfourtrees/choropleth-map-segmentation.git
Cloning into 'choropleth-map-segmentation'...
remote: Enumerating objects: 49, done.
remote: Counting objects: 100% (49/49), done.
remote: Compressing objects: 100% (39/39), done.
remote: Total 49 (delta 11), reused 43 (delta 7), pack-reused 0
Unpacking objects: 100% (49/49), done.
(base) sample_user@demo-machine ~ %

```

Figure 3 – Download the CMS Tool git repository.

- Step 6. Navigate to the cloned project folder with `cd choropleth-map-segmentation`.
- Step 7. Run the `ls` command to list all available folders and files.
- Step 8. Confirm that the folders appear as shown in Figure 4.



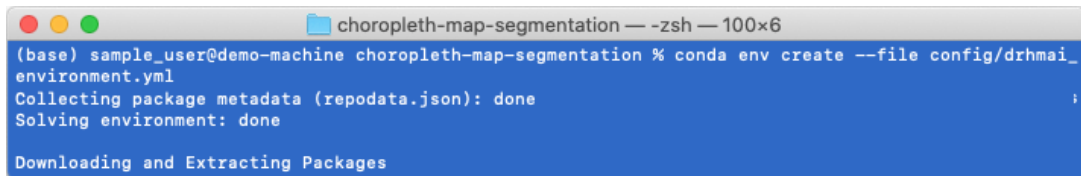
```

choropleth-map-segmentation — -zsh — 100x5
(base) sample_user@demo-machine ~ %
(base) sample_user@demo-machine ~ % cd choropleth-map-segmentation
(base) sample_user@demo-machine choropleth-map-segmentation % ls
README.md      config          input           legend          source
(base) sample_user@demo-machine choropleth-map-segmentation %

```

Figure 4 – Folder structure of the cloned CMS Tool git repository.

- Step 9. Set up the CMS Tool environment by running the command `conda env create --file config/drhmai_environment.yml` as shown in Figure 5.



```

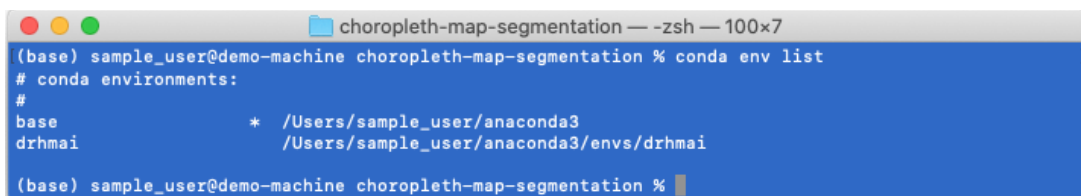
choropleth-map-segmentation — -zsh — 100x6
(base) sample_user@demo-machine choropleth-map-segmentation % conda env create --file config/drhmai_environment.yml
Collecting package metadata (repodata.json): done
Solving environment: done

Downloading and Extracting Packages

```

Figure 5 – Install the conda environment.

- Step 10. Confirm that the environment has been successfully created by running the command `conda env list`. The resulting list should include the `drhmai` environment as shown in Figure 6.



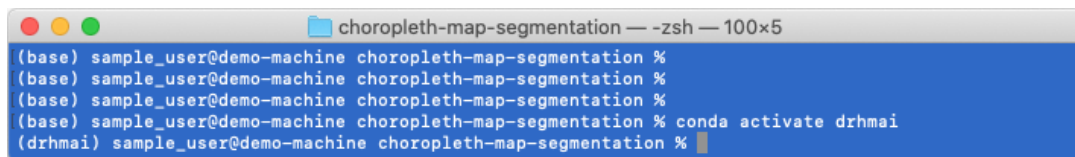
```

choropleth-map-segmentation — -zsh — 100x7
(base) sample_user@demo-machine choropleth-map-segmentation % conda env list
# conda environments:
#
base                * /Users/sample_user/anaconda3
drhmai              /Users/sample_user/anaconda3/envs/drhmai
(base) sample_user@demo-machine choropleth-map-segmentation %

```

Figure 6 – Confirm creation of drhmai environment.

Step 11. Run the command `conda activate drhmai` to use the new environment. The environment name should appear to the left of the command-line terminal window as shown in Figure 7.

A terminal window titled "choropleth-map-segmentation — -zsh — 100x5" with a blue background. It shows a sequence of commands and prompts. The first three lines are "(base) sample_user@demo-machine choropleth-map-segmentation %". The fourth line is "(base) sample_user@demo-machine choropleth-map-segmentation % conda activate drhmai". The fifth line is "(drhmai) sample_user@demo-machine choropleth-map-segmentation %" with a cursor at the end.

```
(base) sample_user@demo-machine choropleth-map-segmentation %  
(base) sample_user@demo-machine choropleth-map-segmentation %  
(base) sample_user@demo-machine choropleth-map-segmentation %  
(base) sample_user@demo-machine choropleth-map-segmentation % conda activate drhmai  
(drhmai) sample_user@demo-machine choropleth-map-segmentation %
```

Figure 7 – Activate the conda environment.

The environment is now installed and ready for running the CMS Tool.

File Management

All files relating to the CMS Tool are stored in the `choropleth-map-segmentation` folder, which contains the following subfolders and a readme file:

`config` – Contains configuration details for installing the anaconda environment (see

- Installation & Setup Step 9) and for styling shapefiles (see Step ###).

`input` – Stores geoTIFF images placed by the user for processing (see

- Preparing Map Files for Input).
- `legend` – Contains images from the original map legend used for analysis of pixel categories. These files are not directly used by the CMS Tool and can be ignored.
- `output` – Stores output for each image processed by the CMS Tool.
- `source` – Contains source code for the CMS Tool and a script to analyze map legend pixels.
- `README.md` – A condensed set of instructions that exists with the project.

The CMS Tool automatically seeks out and reads in any geoTIFF image files stored in the `input` folder. It creates an `output` folder (if one does not already exist) to store the intermediate and final results of the segmentation process.

The `output` folder contains a sub-folder for every image processed by the tool. Each sub-folder contains a log file (`log.txt`) that records key processing parameters along with all process outputs associated with a specific input image. These output files are organized into:

- A `processing` folder.
 - This contains visual evidence of the changes being made to the image file at each step in the algorithm.
 - These files are saved in `.png` format and are numbered sequentially.
- A `segments` folder.
 - This contains the individual map segments, saved as `.tiff` files.
 - The files are named according to their corresponding forest cover category.
 - The category labels are derived from the original map legend for this historical map set.
- A `shapefiles` folder.
 - This contains the map segments in their polygonized form and constitutes the final output of the CMS Tool.
 - Each converted segment has a `.shp`, `.shx`, `.dbf`, and `.cpg` file associated with it.
 - These files can be combined using GIS software to obtain a complete image of the map in a geospatial vector data format if required.

Warning

The `choropleth-map-segmentation` repository (i.e. CMS Tool's project folder) **is not designed to store processing files long-term**.

If the same input image is processed multiple times, all corresponding files in the `output` folder will be overwritten to store the most recent output results. To prevent this, review the output files after each run and copy them to a different location for long-term storage.

Preparing Map Files for Input

The CMS Tool requires input in the form of TIFF images. It is designed to work with map geoTIFFs, but will accept image files with the following extensions:

- .tif or .TIF
- .tiff or .TIFF
- .gtif or .GTIF
- .gtiff or .GTIFF

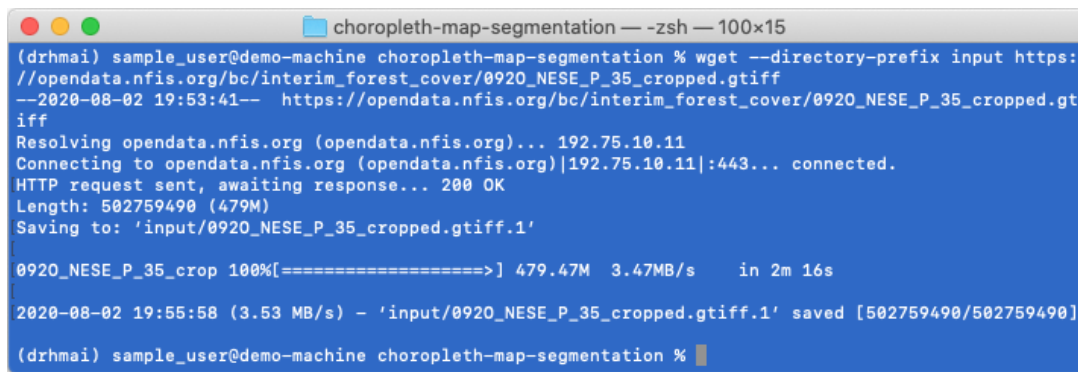
Images can be loaded to the input folder using any of the following ways:

A. Direct Download

- Step 1. Navigate to the `choropleth-map-segmentation` project folder.
- Step 2. Run the command `wget --directory-prefix input <mapURL>`. Be sure to replace `<mapURL>` with a complete URL such as `https://opendata.nfis.org/bc/interim_forest_cover/0920_NESE_P_35_cropped.gtiff`.

This will retrieve a geoTIFF from the NFIS map server and download it directly to the `input` folder as shown in Figure 8.

Note – If `wget` does not work, you may need to install it first.



```

choropleth-map-segmentation — -zsh — 100x15
(drhmai) sample_user@demo-machine choropleth-map-segmentation % wget --directory-prefix input https://opendata.nfis.org/bc/interim_forest_cover/0920_NESE_P_35_cropped.gtiff
--2020-08-02 19:53:41-- https://opendata.nfis.org/bc/interim_forest_cover/0920_NESE_P_35_cropped.gtiff
Resolving opendata.nfis.org (opendata.nfis.org)... 192.75.10.11
Connecting to opendata.nfis.org (opendata.nfis.org)|192.75.10.11|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 502759490 (479M)
Saving to: 'input/0920_NESE_P_35_cropped.gtiff.1'

0920_NESE_P_35_crop 100%[=====] 479.47M  3.47MB/s   in 2m 16s

2020-08-02 19:55:58 (3.53 MB/s) - 'input/0920_NESE_P_35_cropped.gtiff.1' saved [502759490/502759490]

(drhmai) sample_user@demo-machine choropleth-map-segmentation %

```

Figure 8 – Download a map geoTIFF using the command-line interface.

B. Local or Network Copying

- Step 1. Use the `cp` or `scp` commands to copy map geoTIFFs from another location on your machine or network.

It is best to refer to your organization's policies and procedures on using `ssh` and `scp` for remote sharing or copying of files.

Using the CMS Tool

The CMS Tool consists of a Python program that is run using the command-line interface. The tool requests input from the user where required, and provides feedback as it runs through the image segmentation process. This process flow can be represented as a flowchart, as shown in Figure 9.

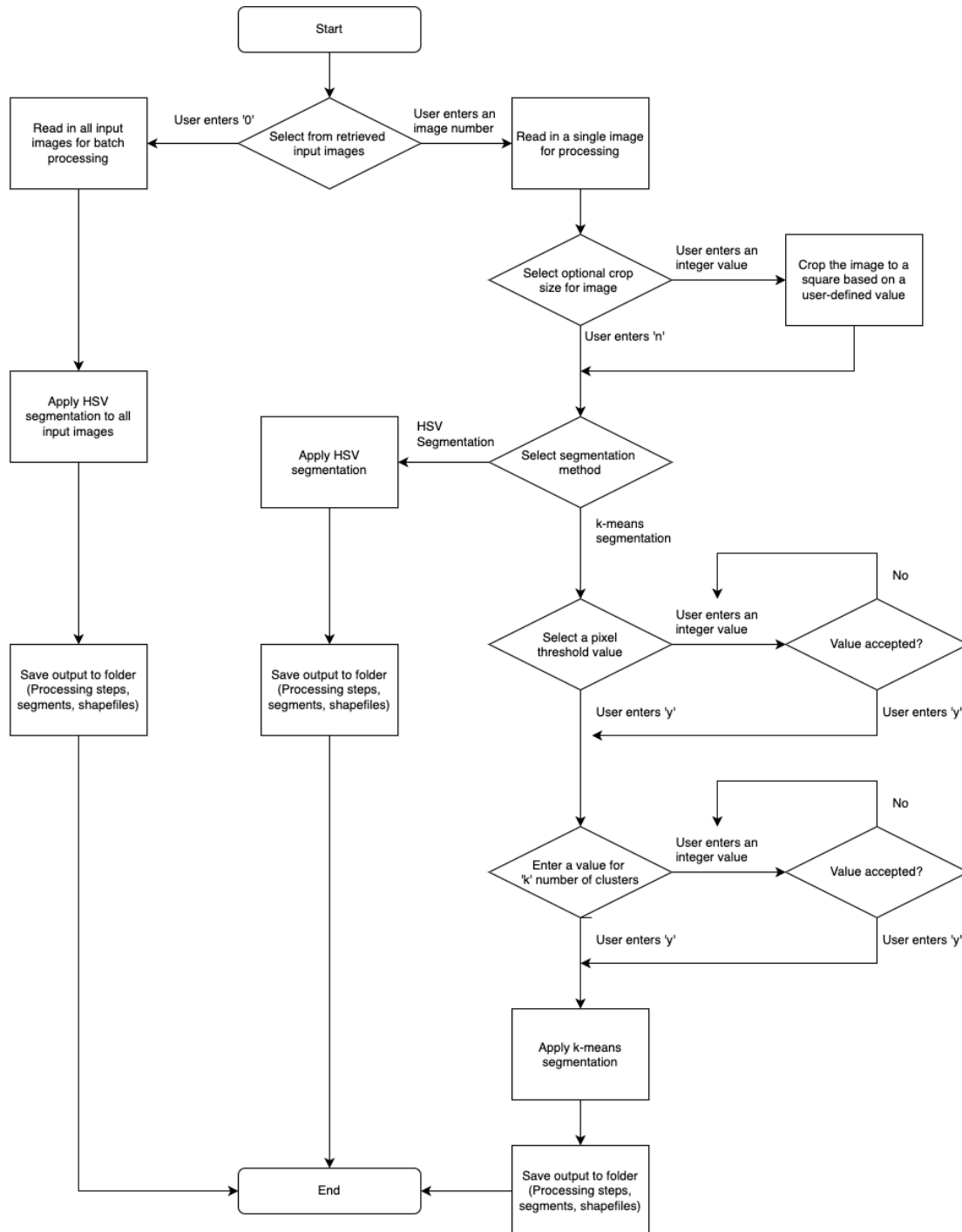
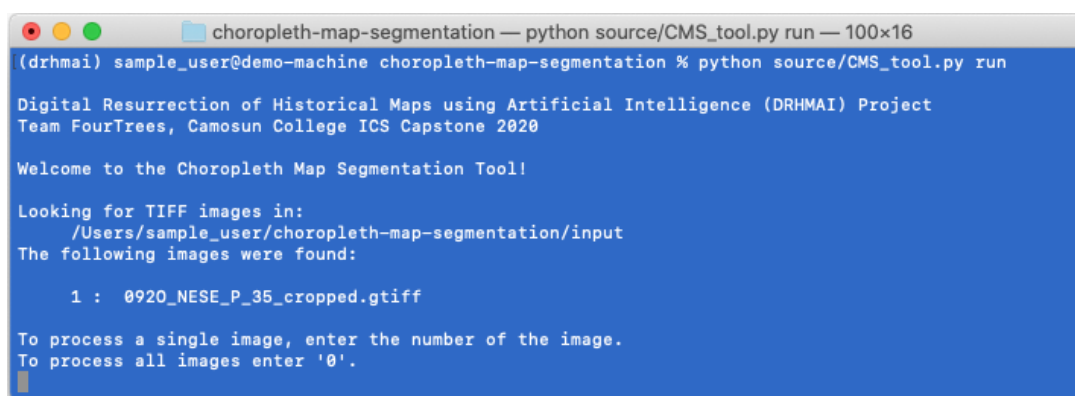


Figure 9 – CMS Tool Process Flowchart.

- Step 1. Change directory to the `choropleth-map-segmentation` folder.
- Step 2. Activate the environment by running `conda activate drhmai`.
- Step 3. Run the command `python source/CMS_tool.py run` to start the program.
You should see a welcome message followed by a list of images retrieved from the `input` directory, as shown in Figure 10.
- Step 4. Read the prompt and make a selection between:
 1. Processing a single image of your choice by entering the corresponding list number.
 2. Processing all the available input images as a batch by entering `0`.

For more details on single image processing, see Steps 5-6. For more details on batch image processing, see Step 7-8.



```

choropleth-map-segmentation — python source/CMS_tool.py run — 100x16
(drhmai) sample_user@demo-machine choropleth-map-segmentation % python source/CMS_tool.py run

Digital Resurrection of Historical Maps using Artificial Intelligence (DRHMAI) Project
Team FourTrees, Camosun College ICS Capstone 2020

Welcome to the Choropleth Map Segmentation Tool!

Looking for TIFF images in:
/Users/sample_user/choropleth-map-segmentation/input
The following images were found:

1 : 0920_NESE_P_35_cropped.gtiff

To process a single image, enter the number of the image.
To process all images enter '0'.

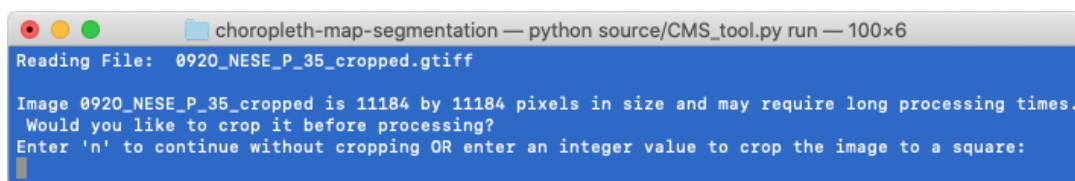
```

Figure 10 – Select the number of input images to process.

A. Single Image Processing Option:

- Step 5. Read the prompt and choose whether you want to crop the map image to a smaller size, as shown in Figure 11.
 1. To ignore this option and proceed with the full image, enter `n`.
 2. To crop the image, enter an integer value (1000 to 3000 is recommended).

This step is useful for testing or building a proof of concept as it can help reduce processing times and output file sizes. However, this method is limited to cropping the shape outward from the upper left corner, and is not intended for specific area cropping.



```

choropleth-map-segmentation — python source/CMS_tool.py run — 100x6
Reading File: 0920_NESE_P_35_cropped.gtiff

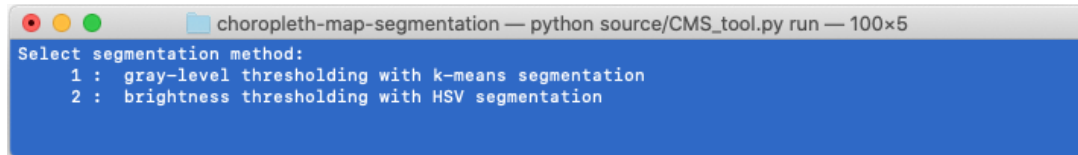
Image 0920_NESE_P_35_cropped is 11184 by 11184 pixels in size and may require long processing times.
Would you like to crop it before processing?
Enter 'n' to continue without cropping OR enter an integer value to crop the image to a square:

```

Figure 11 – Choosing to crop an image.

- Step 6. Read the prompt and choose between segmentation methods, as shown in Figure 12.
 1. To select grayscale-based segmentation with k-means clustering, enter `1`.
 2. To select brightness-based segmentation with hue-saturation-value (HSV), enter `2`.

After you have made your selections, the tool will continue to process the image and provide progress updates in the command-line terminal until it is complete.



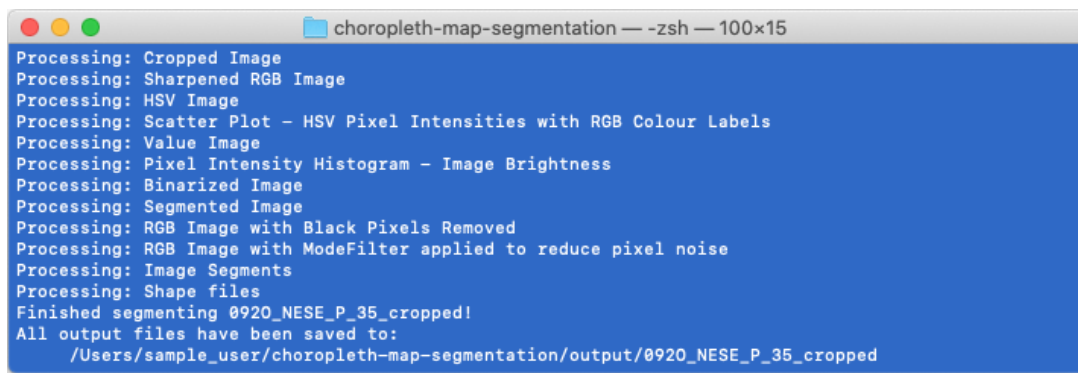
```

choropleth-map-segmentation — python source/CMS_tool.py run — 100x5
Select segmentation method:
 1 : gray-level thresholding with k-means segmentation
 2 : brightness thresholding with HSV segmentation
  
```

Figure 12 – Choosing a segmentation method.

The **HSV segmentation method is recommended** because it outperforms the k-means method and produces more accurate results since it has been fine-tuned to work with the 1950s map set.

If this method is selected, the resulting terminal output will appear as shown in Figure 13.



```

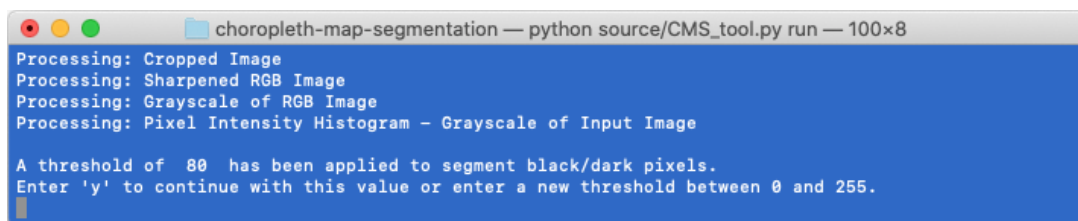
choropleth-map-segmentation — -zsh — 100x15
Processing: Cropped Image
Processing: Sharpened RGB Image
Processing: HSV Image
Processing: Scatter Plot - HSV Pixel Intensities with RGB Colour Labels
Processing: Value Image
Processing: Pixel Intensity Histogram - Image Brightness
Processing: Binarized Image
Processing: Segmented Image
Processing: RGB Image with Black Pixels Removed
Processing: RGB Image with ModeFilter applied to reduce pixel noise
Processing: Image Segments
Processing: Shape files
Finished segmenting 0920_NESE_P_35_cropped!
All output files have been saved to:
/Users/sample_user/choropleth-map-segmentation/output/0920_NESE_P_35_cropped
  
```

Figure 13 – Completing the HSV segmentation and file conversion processes.

The **K-means segmentation option is less effective**, but is provided as it may be useful for testing and experimentation purposes by advanced users. Selecting this option will require additional user input and decision-making.

- 1) Provide a segmentation threshold value between 0 and 255 as shown in Figure 14.

The threshold is used to establish the boundary between dark and non-dark pixels. Typical values range from 70 to 100.



```

choropleth-map-segmentation — python source/CMS_tool.py run — 100x8
Processing: Cropped Image
Processing: Sharpened RGB Image
Processing: Grayscale of RGB Image
Processing: Pixel Intensity Histogram - Grayscale of Input Image

A threshold of 80 has been applied to segment black/dark pixels.
Enter 'y' to continue with this value or enter a new threshold between 0 and 255.
  
```

Figure 14 – Choosing a threshold value.

A histogram plot showing the grayscale pixel intensity distribution will be displayed with a line marking the threshold, as shown in Figure 15. Each time the threshold is adjusted, the histogram is re-plotted and displayed. You can adjust the threshold value multiple times to re-plot and observe its location relative to the distribution.

When you are ready to proceed, enter 'y'.

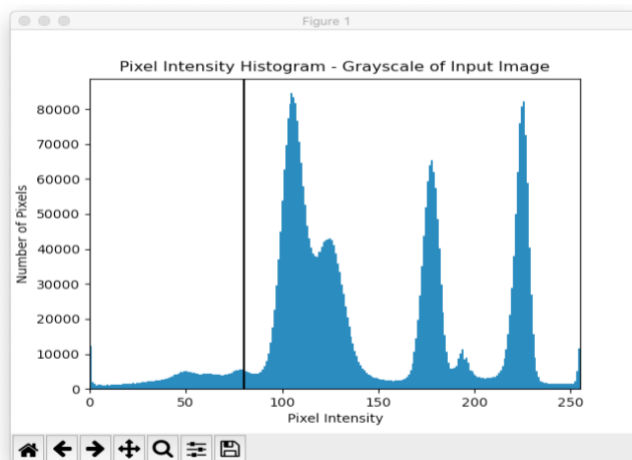


Figure 15 – Observing the threshold value on a pixel intensity histogram plot.

- 2) Enter the number of clusters to use.

This number is the value of 'k' in the k-means algorithm and represents the number of pixel categories on the map. Typical values range from 3 to 7 depending on the map.

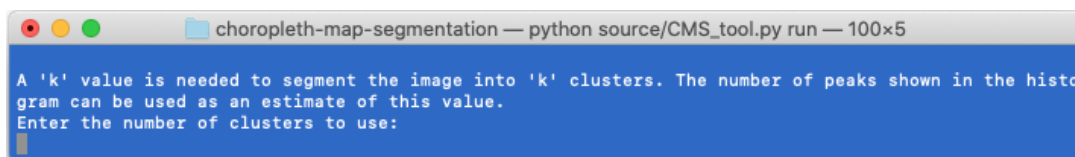


Figure 16 – Choosing a 'k' value for the number of clusters.

- 3) Allow the segmentation process to continue and review the resulting output after the process completion message appears, as shown in Figure 17.

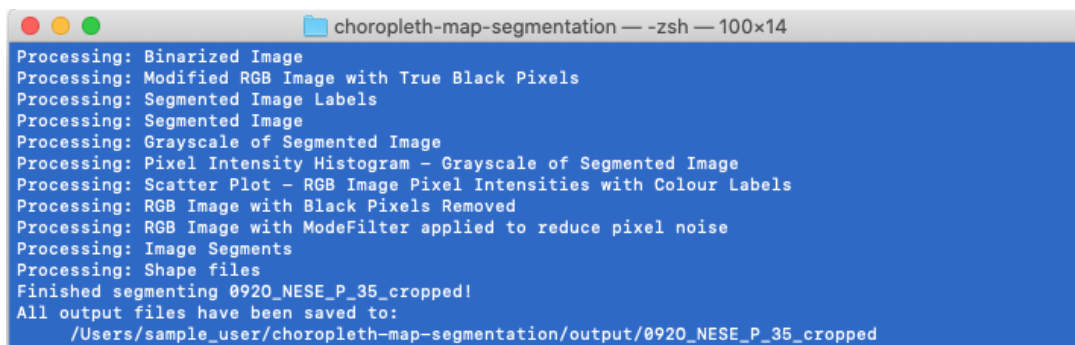


Figure 17 – Completing the k-means segmentation and file conversion processes.

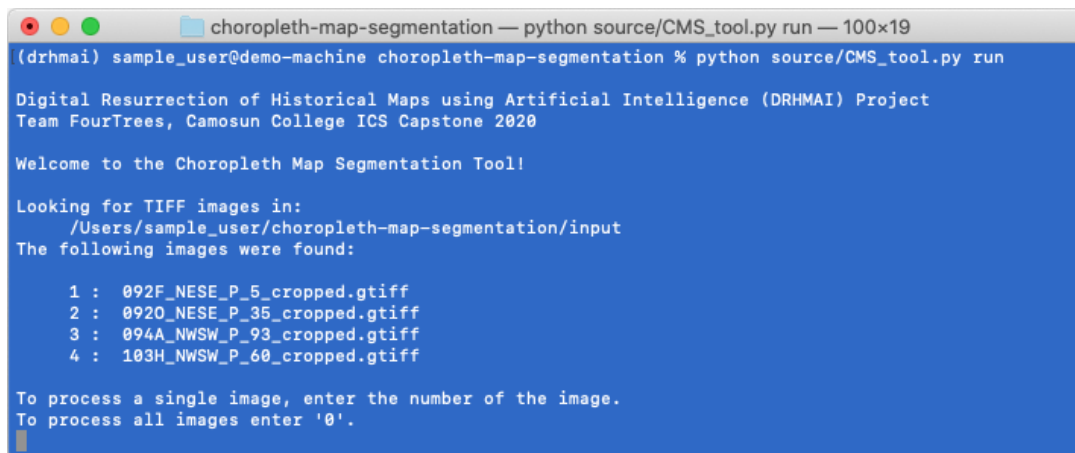
To learn more about the benefits and drawbacks of these two methods, please refer to the **Appendix: K-Means vs HSV Segmentation Methods** section.

B. Batch Image Processing Option:

Step 7. Enter `0` to run the batch image processing option and process multiple files at once as shown in Figure 18.

Please note the following:

- This option will default to using the (recommended) HSV segmentation method.
- This option does not include the ability to crop an image before processing.
- Images are processed sequentially (i.e. as they appear in the input image list).
- Selecting this option may require long processing times per image (1-2 hours or more) depending on the computer's hardware and the file size of each geoTIFF



```

choropleth-map-segmentation — python source/CMS_tool.py run — 100x19
(drhmai) sample_user@demo-machine choropleth-map-segmentation % python source/CMS_tool.py run

Digital Resurrection of Historical Maps using Artificial Intelligence (DRHMAI) Project
Team FourTrees, Camosun College ICS Capstone 2020

Welcome to the Choropleth Map Segmentation Tool!

Looking for TIFF images in:
/Users/sample_user/choropleth-map-segmentation/input
The following images were found:

1 : 092F_NESE_P_5_cropped.gtiff
2 : 0920_NESE_P_35_cropped.gtiff
3 : 094A_NWSW_P_93_cropped.gtiff
4 : 103H_NWSW_P_60_cropped.gtiff

To process a single image, enter the number of the image.
To process all images enter '0'.
0
  
```

Figure 18 – Selecting the batch processing option.

Step 8. If a GUI is available, confirm that the process is complete by navigating to the `output` directory and reviewing the output.

In the absence of a GUI, confirm that output exists by navigating to each of the subfolders in the `output` directory and run the `ls` command to view the enclosed files. For long-term storage and viewing, copy or export the files to a different location. Figure 19, Figure 20 and Figure 21 show the different subfolders within the output folder after processing.

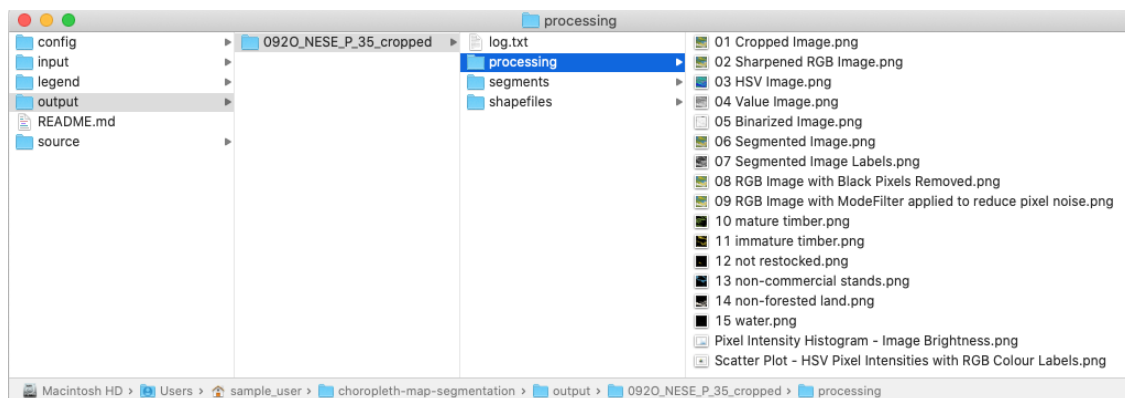


Figure 19 – Example of processing output files after image processing is complete.

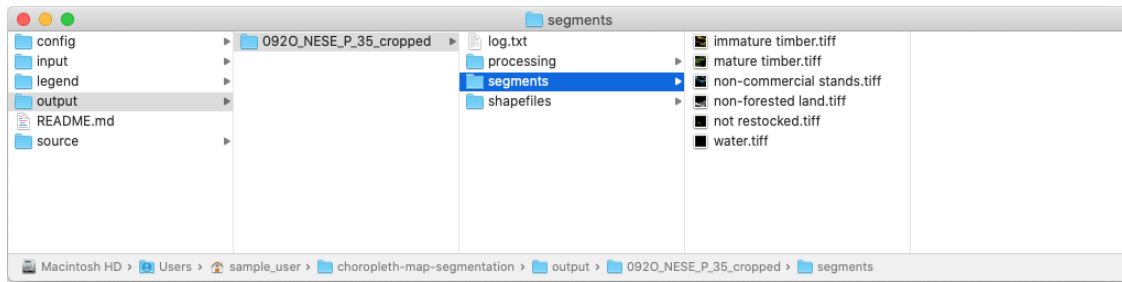


Figure 20 – Example of segment output files after image processing is complete.

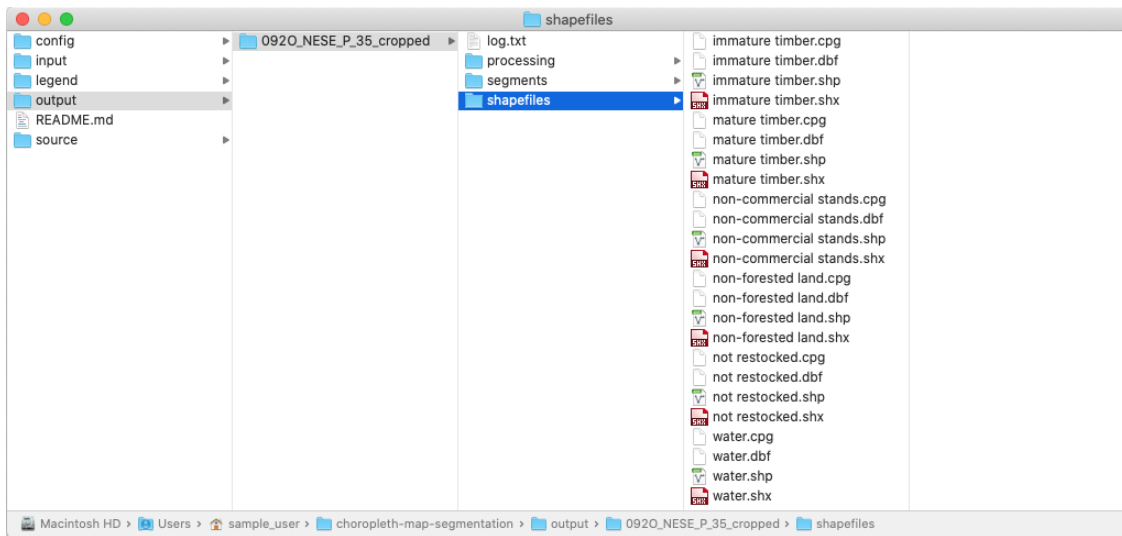


Figure 21 – Example of shapefile output files after image processing is complete.

Example of Segmentation Results

The following graphics present an example of how the CMS Tool modifies and segments an input image (Figure 22) using HSV segmentation (Figure 23) as well as k-means segmentation (Figure 24).

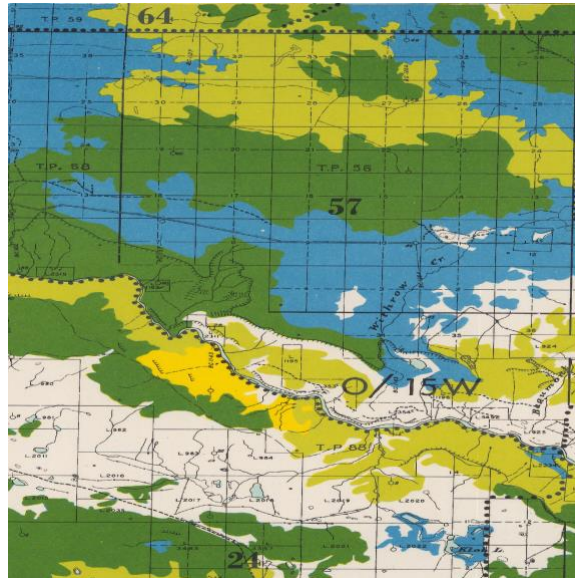


Figure 22 – Example input image 0920_NESE_P_35_cropped.tif cropped to size 2000 x 2000 pixels.

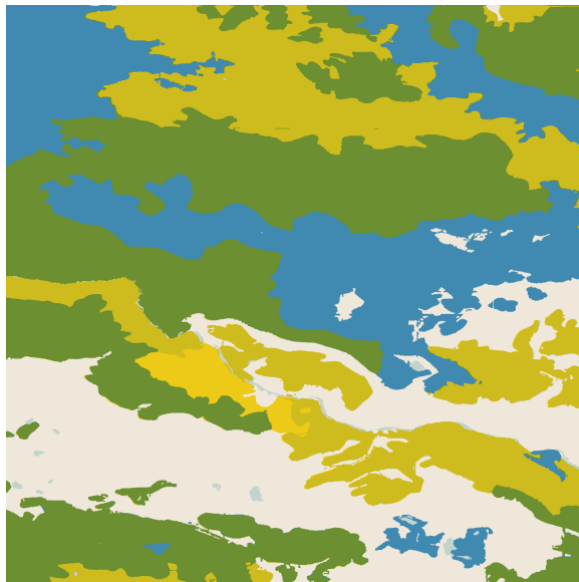


Figure 23 – Example output after HSV segmentation.

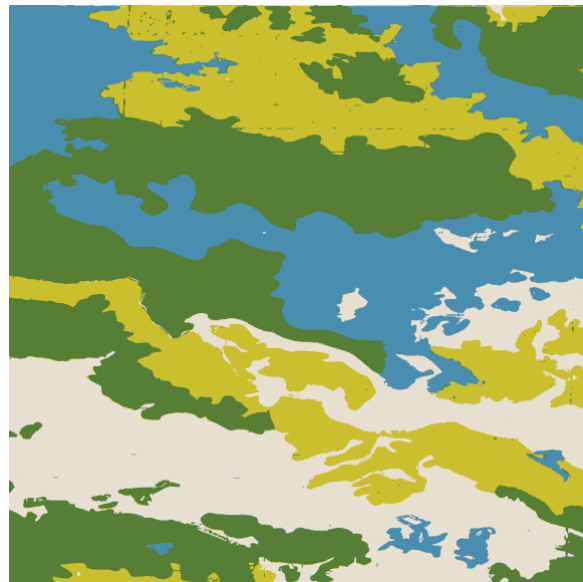


Figure 24 – Example output after k-means segmentation with a threshold of 80 and k-value of 5.

Viewing Shapefiles

Shapefile outputs from the CMS Tool can be viewed using GIS software such as QGIS.

Step 24. Install QGIS. This user manual assumes that version 3.10 is installed.

Step 25. Launch the application.

Step 26. Select “Project->New” from the application menu as shown in Figure 25.

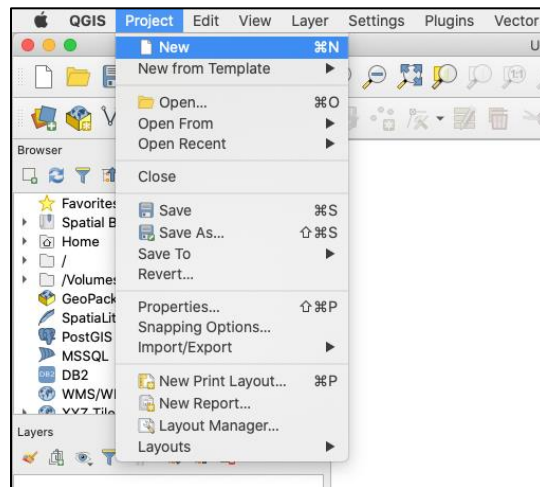


Figure 25 – Create a new project in QGIS.

Step 27. Select “Layer->Add Layer->Add Vector Layer...” as shown in Figure 26.

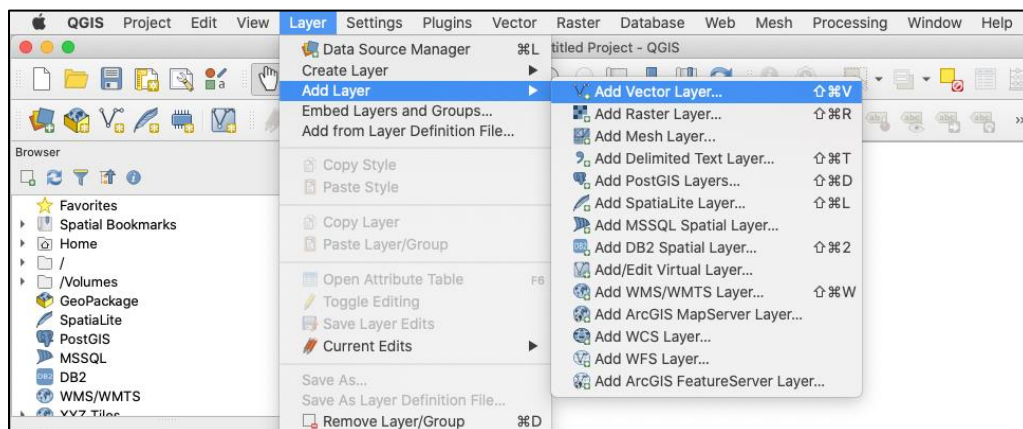


Figure 26 – Add a vector layer.

Step 28. Select the map's shapefile from the CMS Tool's output directory in `~/choropleth-map-segmentation/output/<map>/shapefiles/<map>.shp`. Be sure to substitute `<map>` with a specific map of interest, e.g. `0920_NESE_P_35_cropped` as shown in Figure 27.

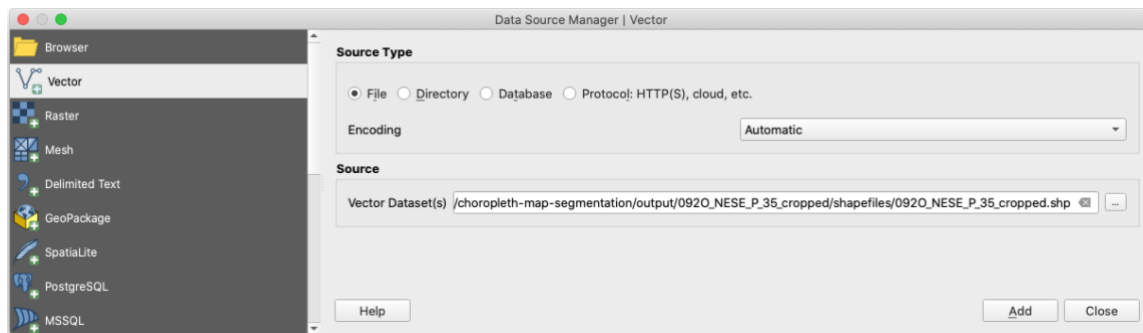


Figure 27 – Select a shapefile to view.

Step 29. Click the “Add” button. The shapefile will appear on the screen as shown in Figure 28.

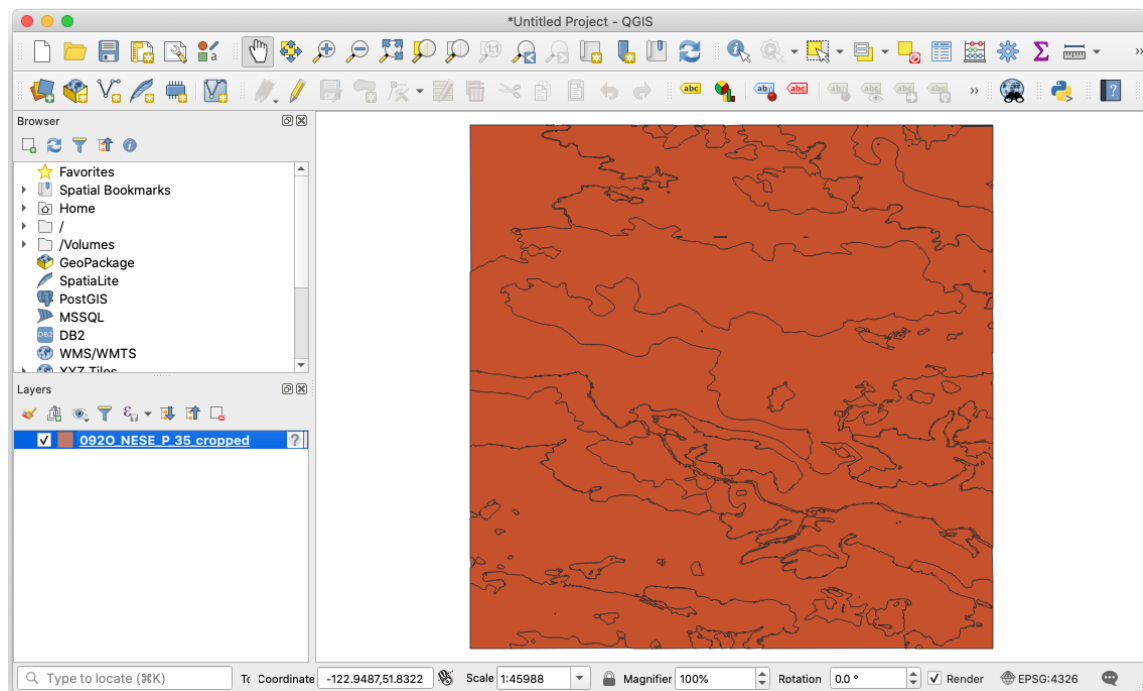


Figure 28 – Map shapefile after being loaded into QGIS.

Step 30. Right-click the new layer from the Layers panel and select “Properties...” as shown in Figure 29.

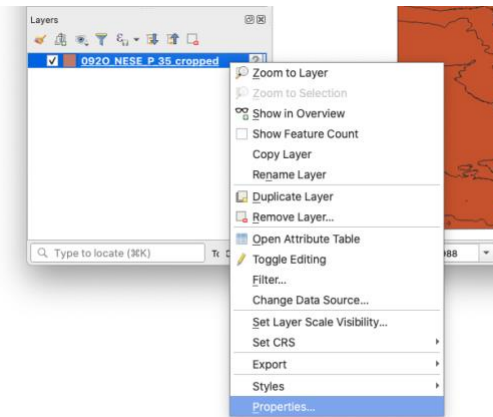


Figure 29 – Navigate to the vector layer properties.

Step 31. Click “Style->Load Style...” from the bottom of the “Symbology” tab as shown in Figure 30.

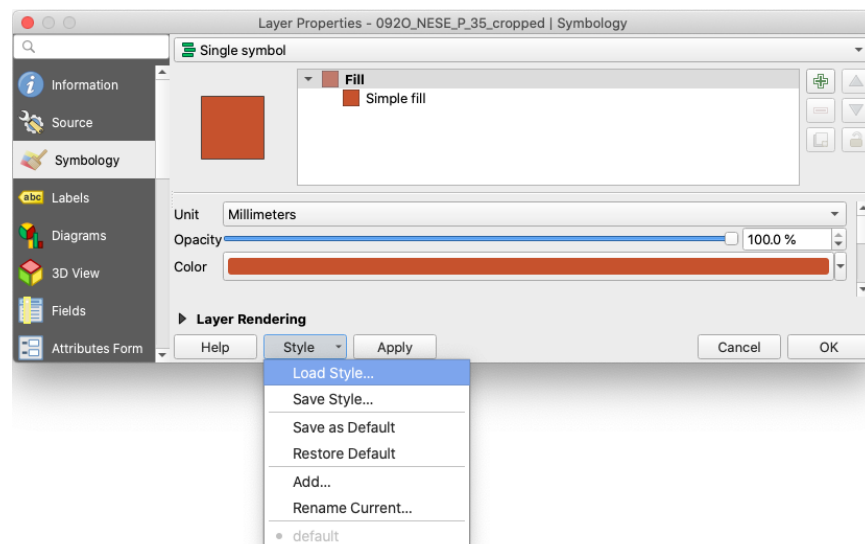


Figure 30 – Navigate to the “Load Style” menu.

Step 32. Select the style `qgis_style.qml` file from the `~/choropleth-map-segmentation/config` folder as shown in Figure 31.

Note – This style is configured to work with HSV-segmented shapefiles only. For shapefiles generated using k-means segmentation a new custom style will need to be created (which is beyond the scope of this user guide).

Step 33. Click “Load Style”. The style will be loaded and the Style Manager will automatically close.

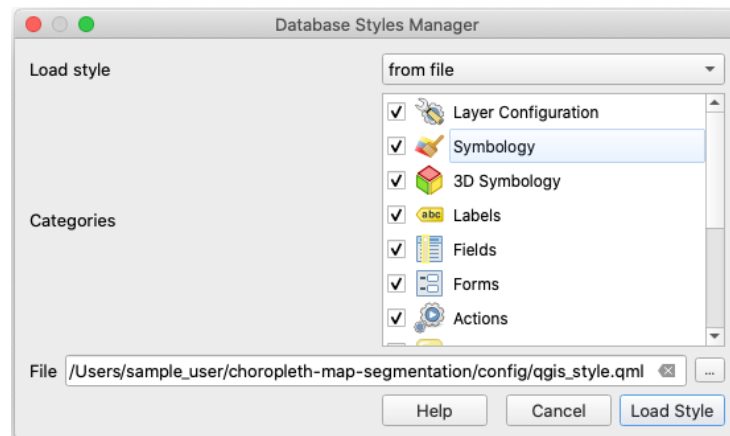


Figure 31 – Load style from file.

Step 34. The “Symbology” tab of the Layer Properties menu will now appear as shown in Figure 32.

Step 35. Click “OK” to close the Layer Properties window.

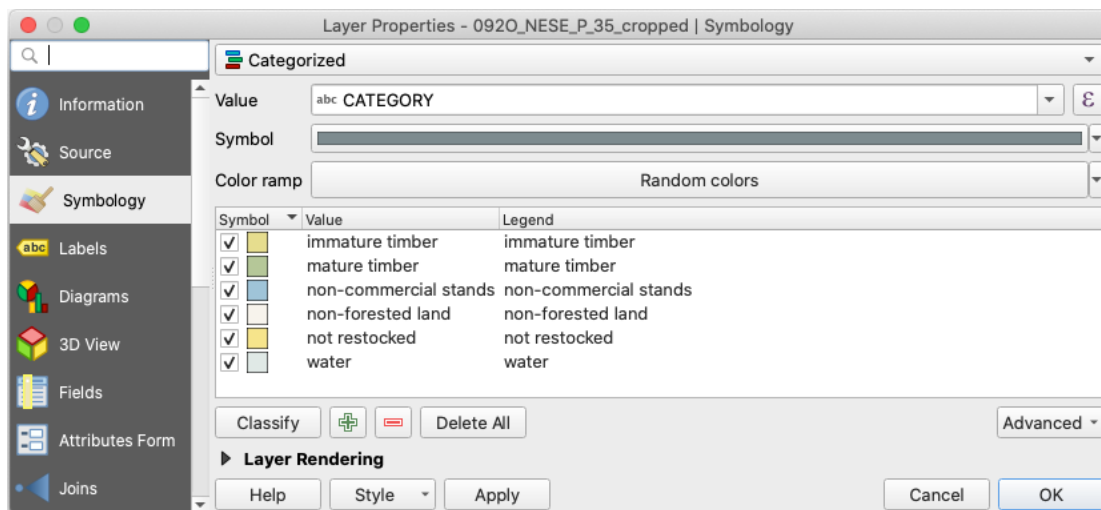


Figure 32 – Symbology tab after loading style file.

The map will appear as shown in Figure 33 with the different forest cover regions coloured and labelled.

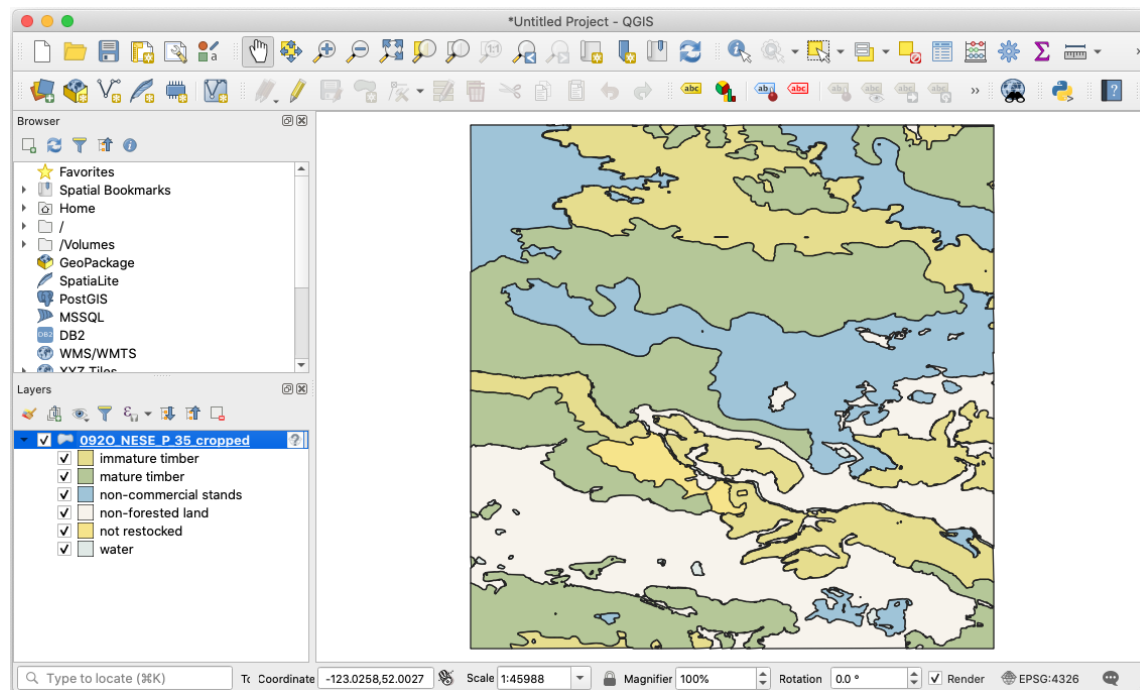


Figure 33 – Map shapefile with style applied.

The shapefile has been successfully loaded and styled and is ready for analysis. To compare the shapefile to the original raster map, proceed with steps ## to ##

Step 36. From the menu, select “Layer->Add Layer->Add Raster Layer...” as shown in Figure 34.

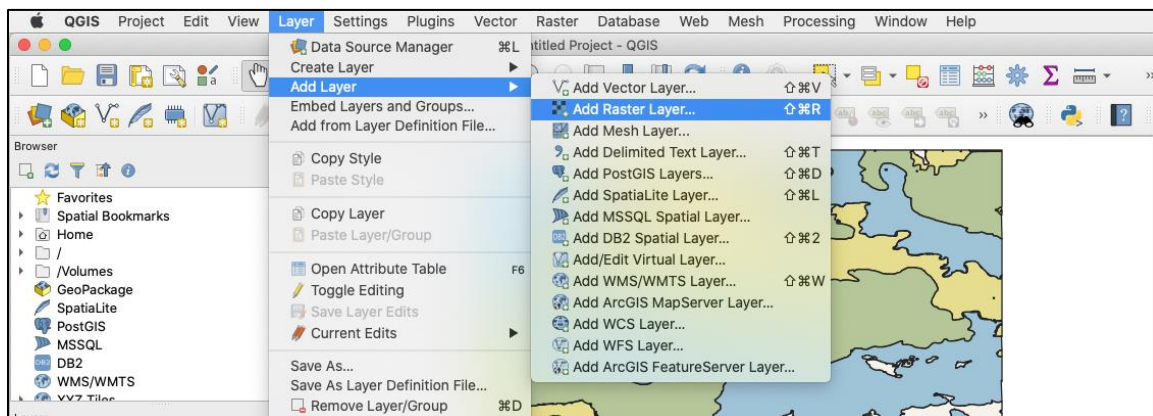


Figure 34 – Add a raster layer.

Step 37. In the “Source type” section, select “Protocol: HTTP(S), cloud, etc.” and enter the map URI, e.g. `opendata.nfis.org/bc/interim_forest_cover/0920_NESE_P_35_cropped.gtiff` as shown in Figure 35.

Alternatively, select the “File” option and select a local file from the “choropleth-map-segmentation/input” directory or elsewhere.

Step 38. Click Add.

Step 39. Click Close. The map should appear in the foreground as a new layer.

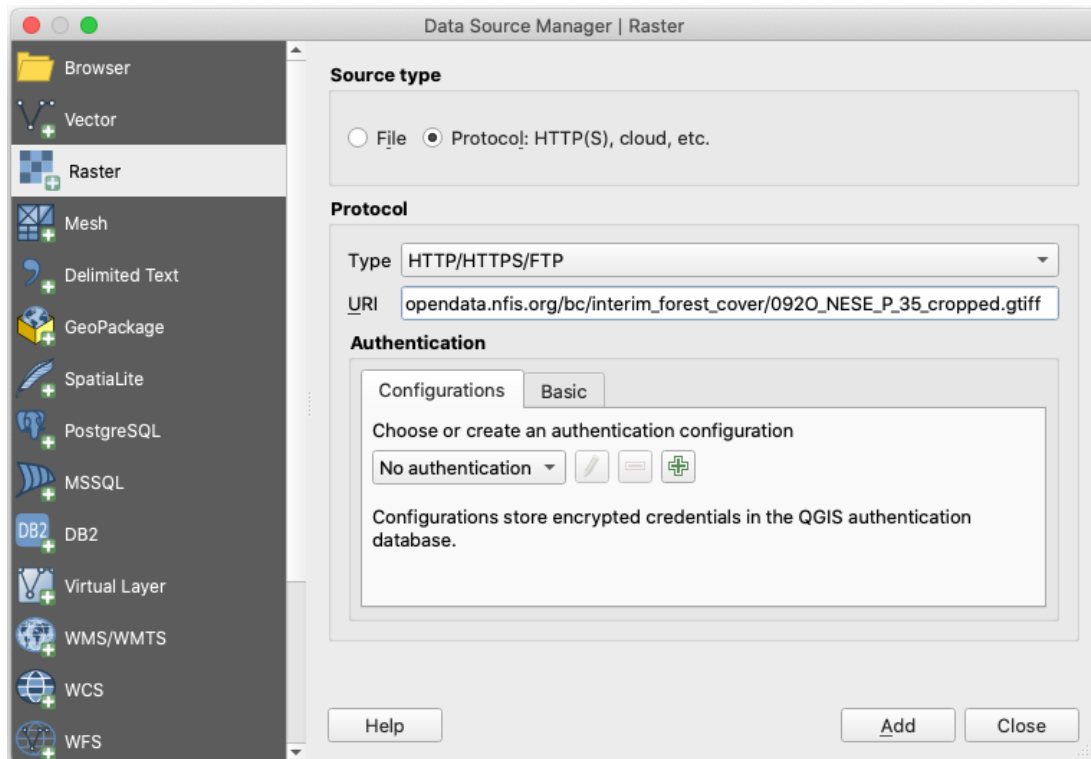


Figure 35 – Select a raster data source.

Step 40. Right-click the new layer from the “Layers” panel and select “Properties...” as shown in Figure 36.

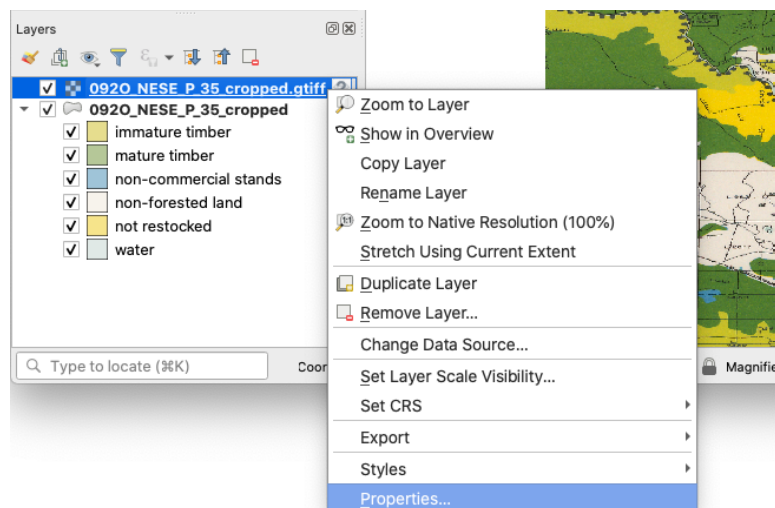


Figure 36 – Navigate to the raster layer properties.

Step 41. In the “Transparency” tab, set the Global Opacity to 50% as shown in Figure 37

Step 42. Click OK.

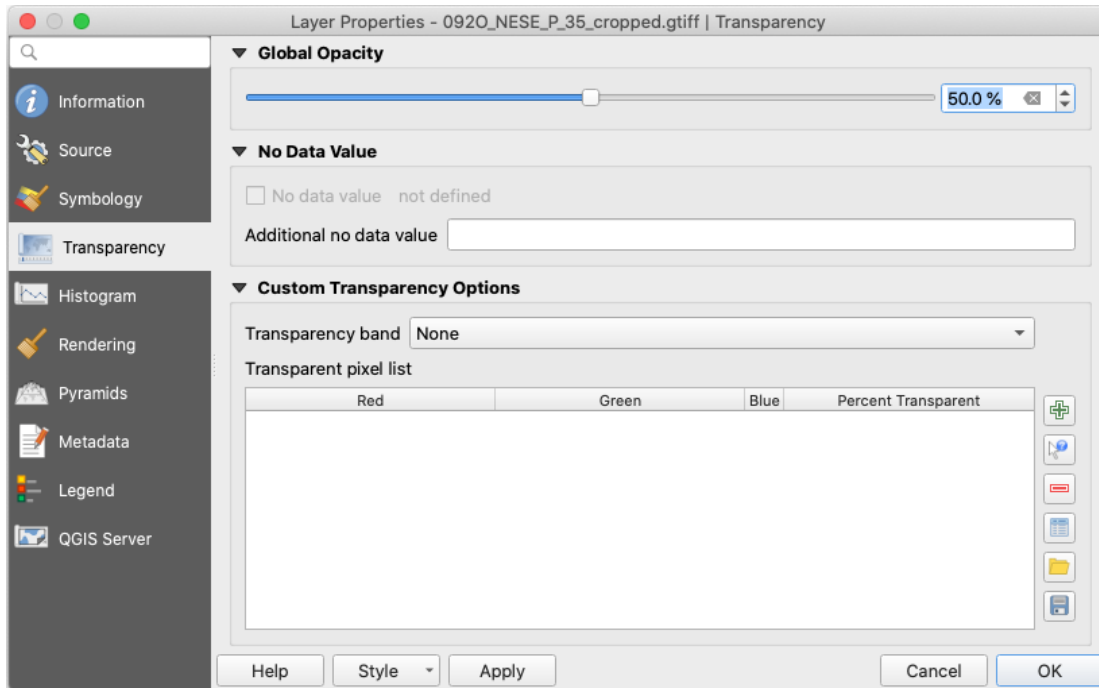


Figure 37 – Modify raster opacity.

The raster image’s colours should now appear to be lighter in colour and blended with the shapefile as shown in Figure 38.

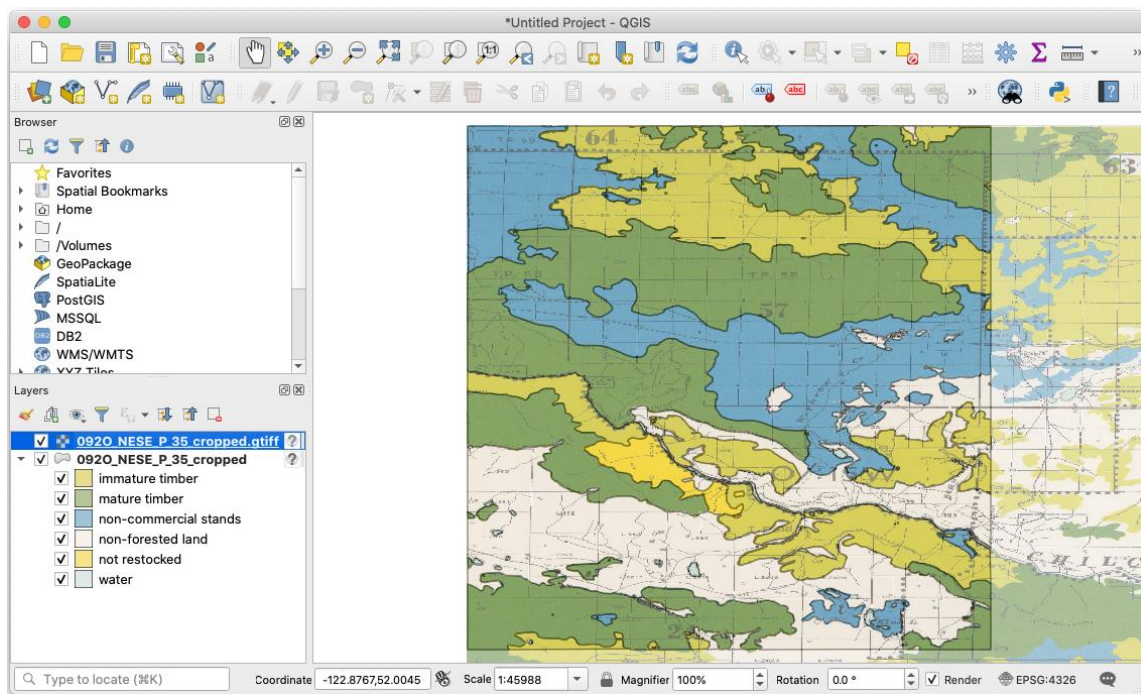


Figure 38 – Map raster and shapefile layers blended together.

- Step 43. Zoom in to a small sub-section of the map, e.g. the section shown in Figure 39.
- Step 44. Click the check-mark to the left of the vector layer in the Layers panel. This will hide it from the view. The map will now appear similar to Figure 40.
- Step 45. Click the check mark to the left of the vector layer again to unhide it.
- Step 46. Click the check-mark to the left of the raster layer in the Layers panel. This will hide it from the view. The map will now appear similar to Figure 41.

By examining the blended view (Figure 39) as well as hiding the and unhiding the raster and vector layers separately (Figure 40, Figure 41) allows the user to make a detailed comparison between the original raster image and the shapefile output from the CMS Tool.

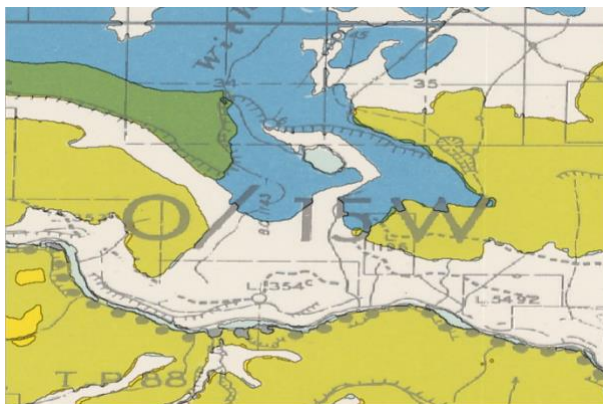


Figure 39 – Detailed map section with raster and vector layers blended together.

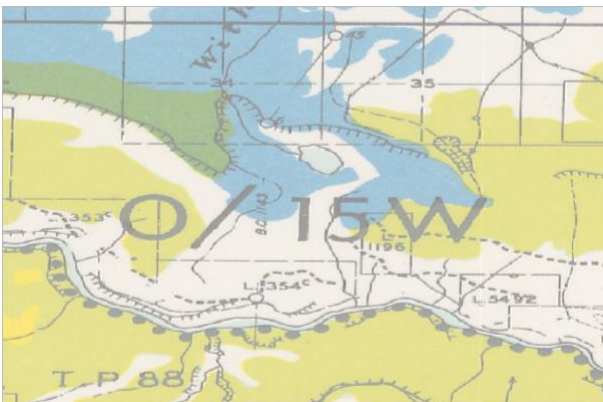


Figure 40 – Detailed map section of the raster layer.

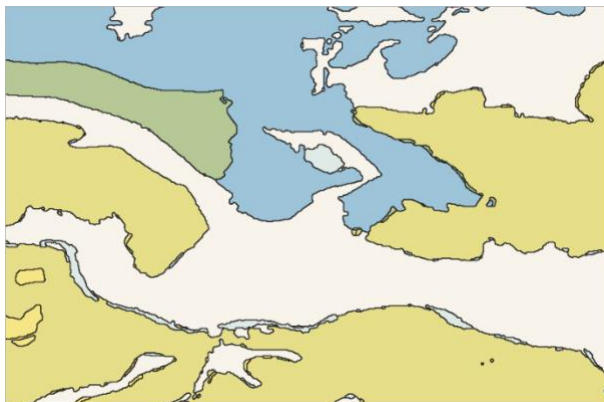


Figure 41 – Detailed map section of the vector layer.

Summary

To use the CMS Tool, you can expect to follow the workflow shown in Figure 42:

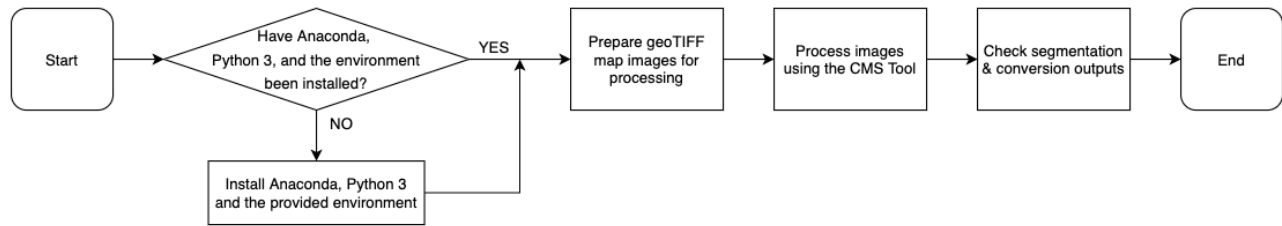


Figure 42 – CMS Tool workflow.

The key steps can be summarized as follows:

1. Prior to using the CMS Tool, install Anaconda and Python 3 onto your machine. Clone the CMS Tool source code from GitHub and activate the provided `conda` environment.
2. Collect map geoTIFFs for processing and place them in the `input` folder. These files will be read in by the CMS Tool.
3. Run the CMS Tool and provide user input when prompted to customize the process.
4. Check the `output` folder to review all image outputs and confirm that the map segments and shapefiles exported properly.
5. The shapefiles can then be viewed using GIS software as needed.

Conclusion

Thank you for using the CMS Tool.

We hope you found this user guide helpful in providing you with an overview of everything this tool can (currently) do.

Should you have any questions about the installation, operation, or functionality of the tool, please feel free to reach out to us at teamfourtrees@protonmail.ch.

Appendix

K-Means vs HSV Segmentation Methods

This table summarizes the key benefits and drawbacks of using K-means or HSV segmentation methods on the provided set of 1950s historical forest cover choropleth maps.

Table 1 – Comparison Summary between K-Means Clustering and HSV Segmentation Methods.

	K-means	HSV
Advantages	<ul style="list-style-type: none"> • Semi-automatic Unsupervised machine learning: The pixel colour classes are determined through comparison between the colour value of a central pixel with all the pixels surrounding it within a specific cluster. • The user has the ability to set a custom threshold value and k-value to best fit their segmentation needs. • This is useful for experimentation and testing, and can be used with more than one type of choropleth map. 	<ul style="list-style-type: none"> • Outperforms k-means on segmentation of large map geoTIFF images. • Ability to fine-tune upper and lower bounds for pixel classes provides greater probability of accuracy. • A good option if a legend or classification scheme is provided (i.e. the classes are known in advance). • Does not require input from the user which could be beneficial as it abstracts away complexity.
Disadvantages	<ul style="list-style-type: none"> • Map images have to be cropped for successful segmentation results as k-means works better on smaller images. • Segmentation is inaccurate for large historical map images, likely because of high pixel colour variation, which increases the risk of pixel misclassification. • Requires the user to have some knowledge of how k-means works to use it effectively 	<ul style="list-style-type: none"> • Colour boundaries have to be manually defined, which means the algorithm has to be modified with new colour data if a different map set is being processed. • Maps that are not compatible with the pre-defined legend may cause inaccuracies with segmentation.