

# **HSItools documentation**

Maurycy Żarczyński

# Table of contents

<b>Preface</b>	<b>4</b>
Installation . . . . .	4
QGIS . . . . .	5
<b>1 Initial state</b>	<b>6</b>
1.1 Data structure . . . . .	6
1.2 Startup . . . . .	7
<b>2 Shiny app</b>	<b>8</b>
2.1 Screen 1: Initial settings . . . . .	8
2.2 Screen 2: Data choice . . . . .	9
2.3 Screen 3: Cropping . . . . .	14
2.4 Screen 4: ROI selection . . . . .	15
2.5 Screen 5: Calibration . . . . .	16
<b>3 Preprocessing</b>	<b>19</b>
3.1 Normalization . . . . .	19
3.1.1 Normalization with Shiny output . . . . .	20
3.1.2 Normalization of the directory (no Shiny output) . . . . .	20
3.2 Savitzky-Golay smoothing . . . . .	21
3.3 Median filter . . . . .	22
3.4 Continuum removal . . . . .	22
<b>4 Regions of interest</b>	<b>23</b>
4.1 Short path – The Shiny Path . . . . .	23
4.2 Long path – The GIS Path . . . . .	23
<b>5 Indices</b>	<b>24</b>
5.1 Mean reflectance (Rmean) . . . . .	24
5.2 Relative Absorption Band Depth (RABD) . . . . .	24
5.2.1 Variant 1 – “max” . . . . .	24
5.2.2 Variant 2 – “strict” . . . . .	25
5.2.3 Variant 3 – “midpoint” . . . . .	25
5.3 Relative Absorption Band Area - ToDo . . . . .	26
5.4 Spectral ratios . . . . .	26
5.5 Derivatives - ToDo . . . . .	27

5.6	Differences . . . . .	27
5.7	Other . . . . .	28
5.7.1	Red Edge Minimum Point (REMP) - ToDo . . . . .	28
<b>6</b>	<b>Visualizing the data</b>	<b>29</b>
6.1	RGB preview . . . . .	29
6.2	Index map . . . . .	29
6.3	Index plot . . . . .	29
6.4	Spectrum plot . . . . .	29
<b>7</b>	<b>Extracting the data</b>	<b>30</b>
<b>8</b>	<b>Summary</b>	<b>31</b>
	<b>References</b>	<b>32</b>

# Preface



This is a companion book to the [HSItools](#) R package, aiming at processing and visualizing hyperspectral scanning data.

Maurycy Żarczyński, David C. Edge, Nick P. McKay, and Paul D. Zander developed the package with the community's help.

The current requirements to run HSItools are as follows:

- R:  $\geq 4.1.0$  is necessary because we depend on the native R pipe and lambda functions introduced with R 4.1.0.
- Rtools (on Windows):  $\geq 4.0$  [\[link\]](#)
- QGIS (optionally but heavily encouraged): [\[link\]](#)

These downloads and installation take some time, depending on the Internet speed and your Operating System.

## Installation

{HSItools} is currently in development and available only from GitHub.

### 💡 Package installation

We are using {pak} for installation from remote sources.

```
# Install pak if necessary
install.packages("pak")

# Install HSItools
pak::pak("mzarowka/HSItools")
```

! Terra

Because the latest version of {terra}, which does most of the raster heavy-lifting, has a bug resulting in flipped images, we must install older version.

```
# Install older version from CRAN
pak::pak("terra@1.7-78")
```

## QGIS

In our workflow we delegate some of the work to GIS environment, especially marking the raster files, selecting regions and so on. We strongly suggest an Open Source tool like QGIS, however any GIS software, such as ESRI ArcGIS Pro should suffice.

You can find the latest version of QGIS here: <https://www.qgis.org>

# 1 Initial state

HSItools offers an easy way to preprocess Specim data. However, if the data follows the same rules, it can be generalized to the broader workflow.

## 1.1 Data structure

Data should be structured as follows:

```
- NAME <directory>
  -- capture <directory>
    --- DARKREF_NAME.hdr
    --- DARKREF_NAME.log
    --- DARKREF_NAME.raw
    --- NAME.hdr
    --- NAME.log
    --- NAME.raw
    --- WHITEREF_NAME.hdr
    --- WHITEREF_NAME.log
    --- WHITEREF_NAME.raw
```

However, if necessary, you can select appropriate files on your own. Files with the extension .raw are data files, while files with the .hdr extension are header files that contain essential information, such as the number of pixels, facilitating data reading.

## 1.2 Startup

Start by loading all necessary packages{HSItools} will load essential functions from the namespace, but to be sure, call all packages.

```
# Load packages
library(HSItools)
library(terra)
library(tidyterra)
library(dplyr)
library(tidyr)
library(ggplot2)
library(signal)
library(sf)
```

Hyperspectral data gets quite large. It is a good practice to process data stored on an SSD drive separate from your Operating System (OS). While data is stored on a separate drive, it is beneficial to instruct {terra} to store temporary data on a separate drive, too. Adjust this according to your OS. Here, we are using a non-system drive on Windows 11. We set the maximum allowed RAM to a high value of 0.9. It would be best to decide according to the available memory and OS requirements.

```
# Set tempdir
terra::terraOptions(tempdir = "D:/", memmax = 0.9)
```

## 2 Shiny app

Our shiny app allows quick choice of data, settings, regions of interest (ROI), and depth calibration. Here, we walk through the entire app, screen by screen.

### 2.1 Screen 1: Initial settings

On this screen, you have to make an initial choice. First, you need to decide whether to normalize data. If you have a reflectance file from other software, like Lumo®, you probably do not need to normalize the file from the beginning—this is one of the most time-consuming processes.

Suppose you decide that you need to normalize your data. In that case, you can select other integration times for your white and dark references if you scanned your target with different settings for target and references. This can happen if you were worried about white reference overexposure, whereas your target was very dark. Finally, you can select some proposed HSI indices from the defaults.

#### Indices

At the moment, selecting indices will not do anything in particular.

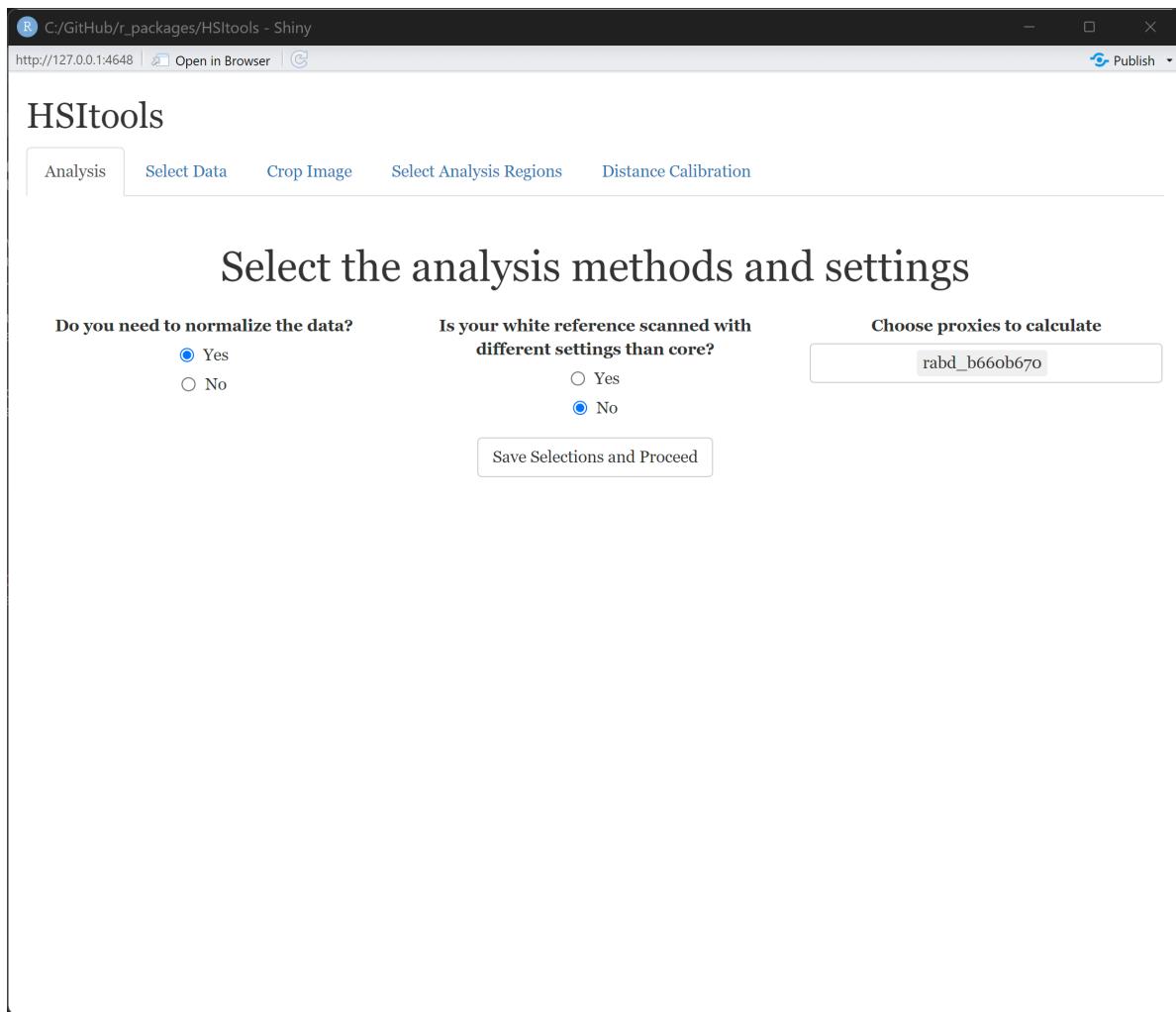


Figure 2.1: Shiny – first screen, basic options.

## 2.2 Screen 2: Data choice

This screen lets you decide whether to use your data or the provided example.

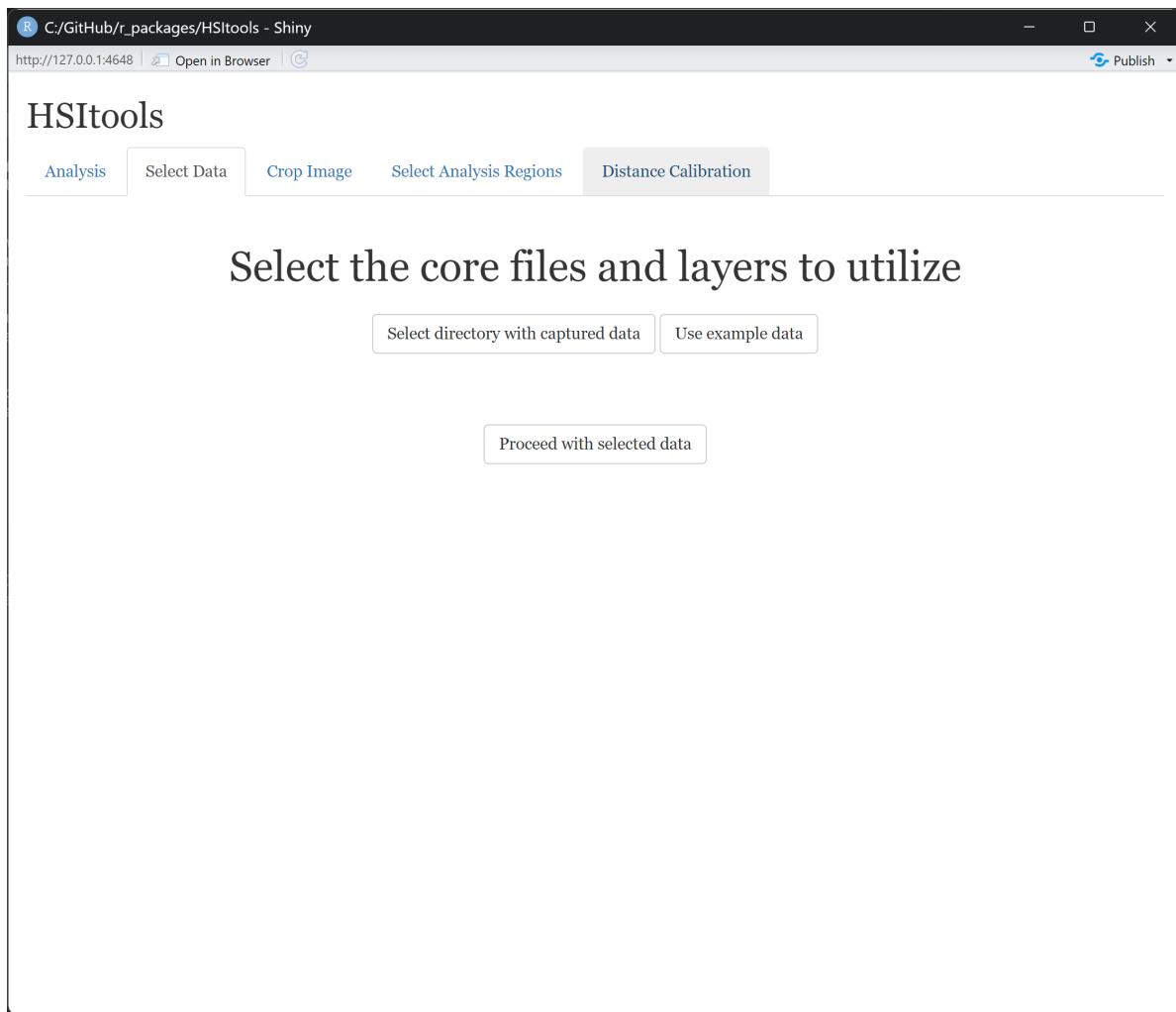


Figure 2.2: Shiny – data choice.

You can select your data from your drive once you use it. Be sure that it is stored on the SSD drive. Here, it is essential that you do not go all the way to the capture folder but rather point to the root directory of your selected scan.

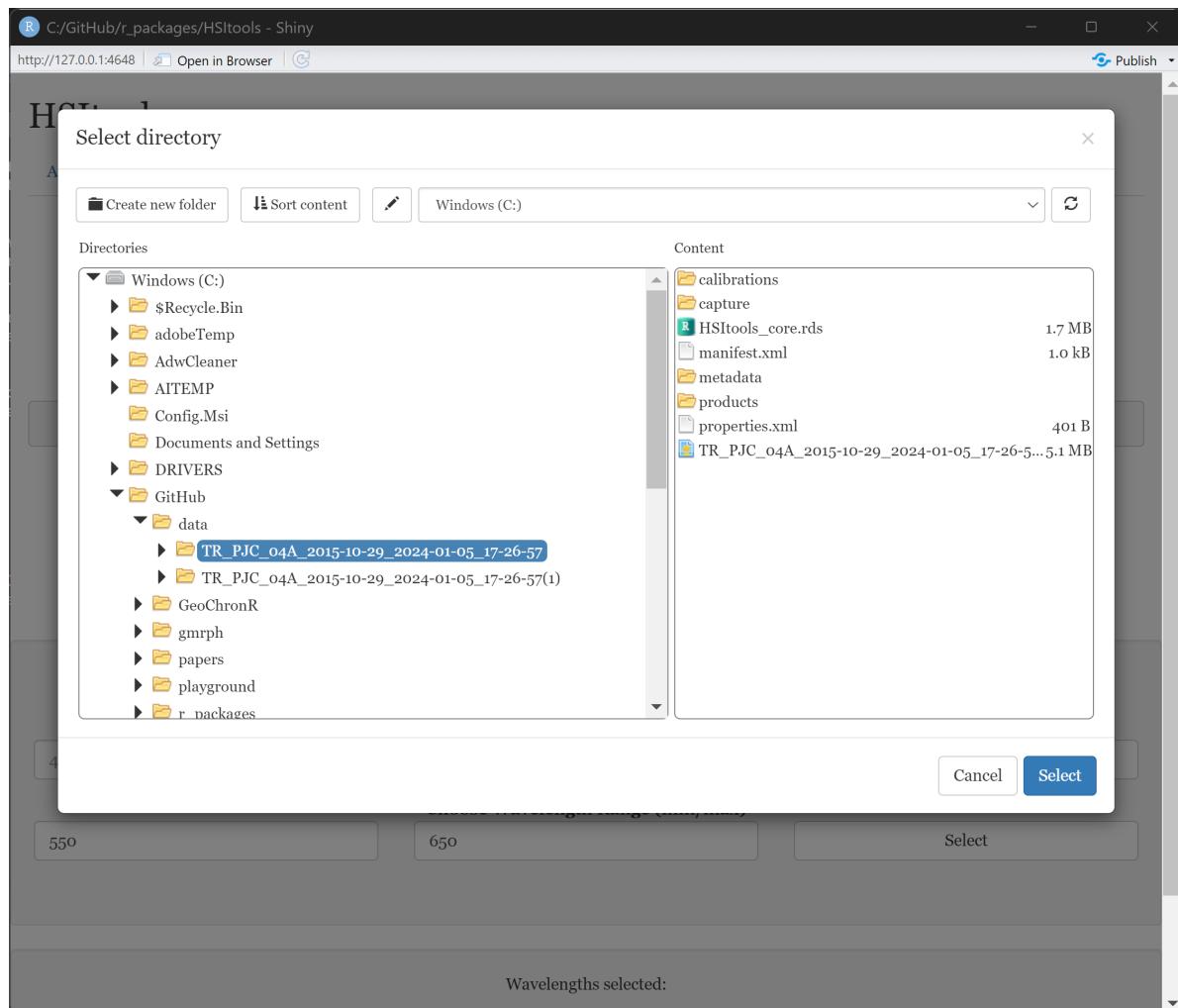


Figure 2.3: Shiny – data selection.

Once you've selected data, you can choose the raster files, bypassing our heuristics and default search patterns in the “capture” directory. At this point, you'll also get some basic information on your raster dimensions.

The screenshot shows the HSItools Shiny application interface. At the top, there is a navigation bar with tabs: Analysis, Select Data (which is currently selected), Crop Image, Select Analysis Regions, and Distance Calibration. Below the navigation bar, the title "HSItools" is displayed. A main heading "Select the core files and layers to utilize" is centered. Two buttons are present: "Select directory with captured data" and "Use example data". A section titled "Selected core directory" shows the path "C:/GitHub/data/TR\_PJC\_04A\_2015-10-29\_2024-01-05\_17-26-57". Below this, three dropdown menus are shown under the heading "Raster files in the directory": "Image" (containing "TR\_PJC\_04A\_2015-10-29\_2024-01-05\_17-26-57"), "White Reference" (containing "WHITEREF\_TR\_PJC\_04A\_2015-10-29\_2024-01-05"), and "Dark Reference" (containing "DARKREF\_TR\_PJC\_04A\_2015-10-29\_2024-01-05"). Each dropdown has a "Change Selection" button below it. To the right of these dropdowns, a box displays file metadata: width: 1312 pixels, height: 3389 pixels, layers: 392. Below this, a section titled "Choose layers to subset raster" contains two checkboxes: "select/deselect all" (checked) and "select every other row" (unchecked). A text input field "Choose Custom Wavelengths (comma separated)" contains the value "450,550,650" and a "Select" button. A final section titled "Choose Wavelength Range (min/max)" is partially visible at the bottom.

Figure 2.4: Shiny – data selection continued.

Once you're happy with your data choice, select layers (bands) to work with.

The screenshot shows a Shiny application window titled "Choose layers to subset raster". At the top, there are two checkboxes: "select/deselect all" (checked) and "select every other row". Below this is a section titled "Choose Custom Wavelengths (comma separated)" with an input field containing "450,550,650" and a "Select" button. Another section titled "Choose Wavelength Range (min/max)" has input fields for "550" and "650" and a "Select" button. Below these controls is a table with columns "index" and "wavelength". The table contains 10 rows of data, indexed from 1 to 10. The "wavelength" column values are: 395.5, 397, 398.49, 399.99, 401.49, 402.99, 404.48, 405.98, 407.48, 408.98. At the bottom left, it says "Showing 1 to 10 of 392 entries". At the bottom right, there are navigation buttons for "Previous" (disabled), "1", "2", "3", "4", "5", "...", "40", and "Next". A message at the bottom states "Wavelengths selected: 395.50, 397.00, 398.49, 399.99, 401.49, 402.99, 404.48, 405.98, 407.48, 408.98, 410.48, 411.98, 413.48, 414.98, 416.49, 417.99," followed by a dropdown arrow.

	index	wavelength
1	1	395.5
2	2	397
3	3	398.49
4	4	399.99
5	5	401.49
6	6	402.99
7	7	404.48
8	8	405.98
9	9	407.48
10	10	408.98

Figure 2.5: Shiny – data selection, layers.

If you're happy with your choices, scroll down and proceed.

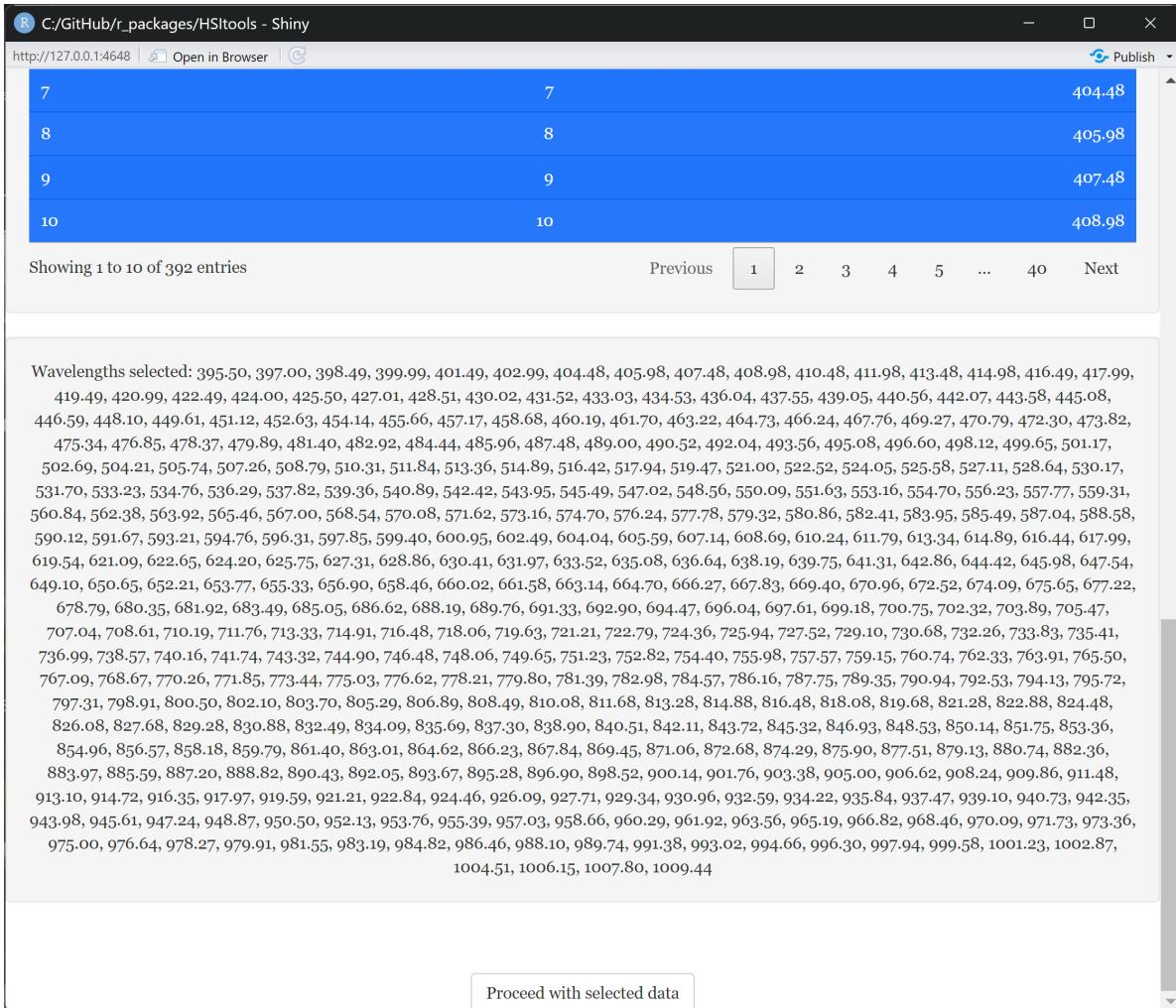


Figure 2.6: Shiny – data selection, finish.

## 2.3 Screen 3: Cropping

**!** Raster preview

Some scans will look “bad” on this and the next screen. It is still your data. We’re working on it! :)

Here, you can crop your data (raster). You can see the bounding box of the cropped data. If it helps, you can change the size of the image. Leave a pixel or two on the edges; if pushed to the extremes, it can mess up the cropping. Once you’re happy, accept the crop and proceed.

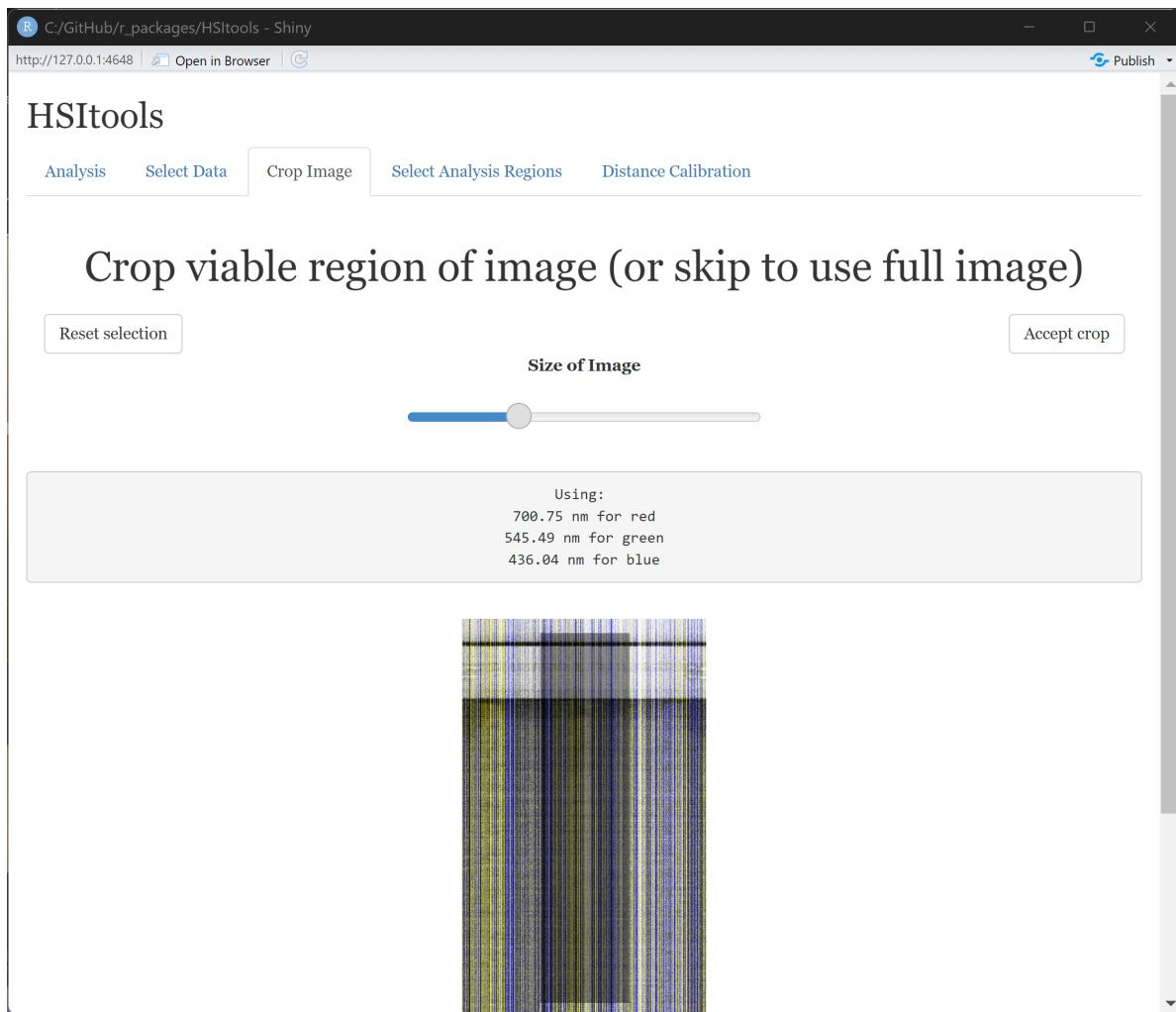


Figure 2.7: Shiny – cropping.

## 2.4 Screen 4: ROI selection

Here, you can select the regions of interest (ROI) you want to work over rather than calculating the entire data. These ROIs are strictly rectangular, so we prefer the GIS approach with masking. Please select the region, then add it using the button. Repeat for as many regions as you need. On the right, you have a zoom of your selection. Once you're happy with your ROIs, accept and proceed.

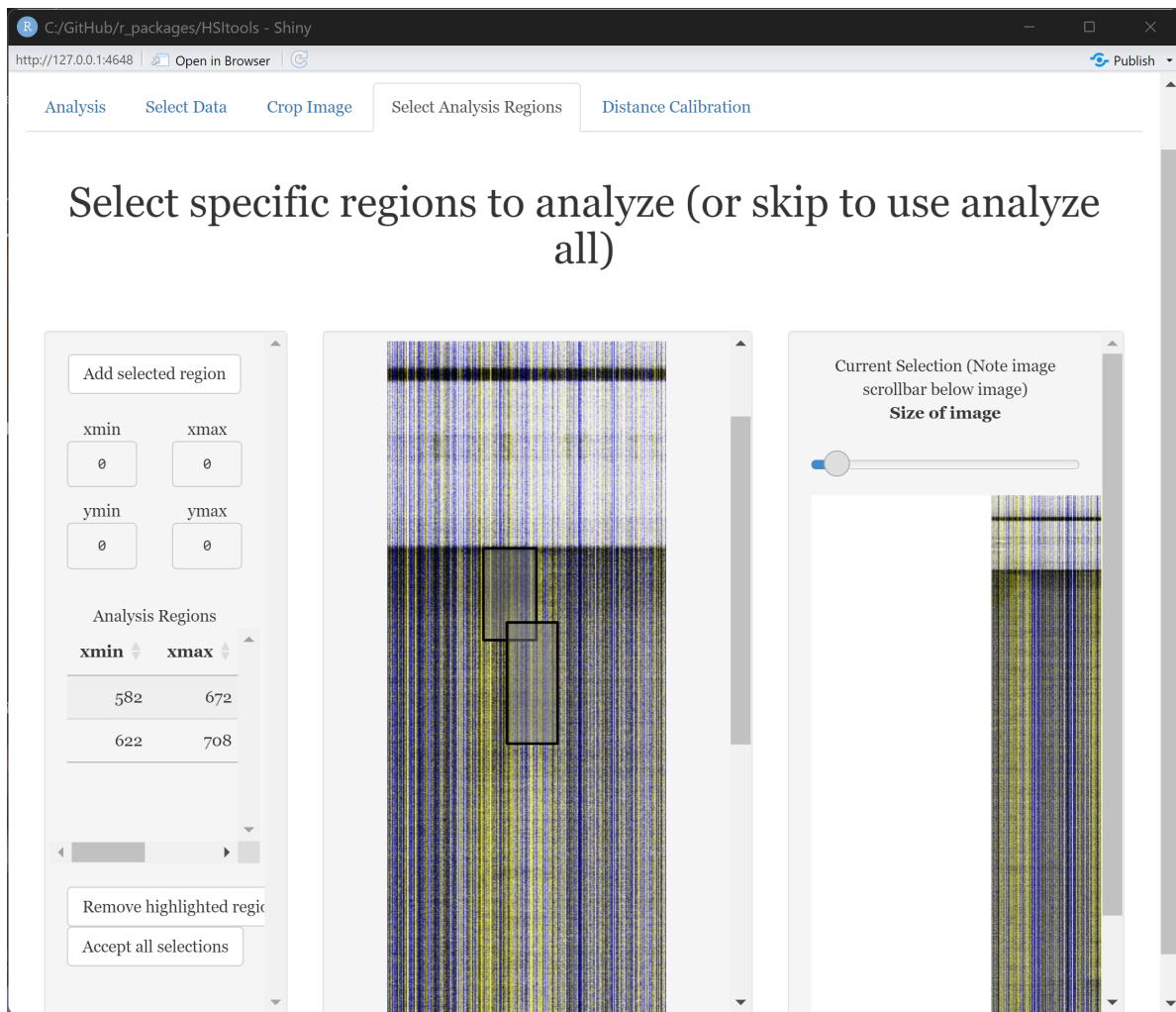


Figure 2.8: Shiny - ROIs selection.

## 2.5 Screen 5: Calibration

! RGB preview

Here, the captured data looks as it should!

Finally, it is time to calibrate your data.

First, we calibrate the distance. Simply put, we want to know the actual distance in one pixel. For this, we use the tape measure and put the two markers. Then, we read the depth from

the tape—the further the distance between the points, the better. We use only 20 mm, so it fits on the screen. Our equation takes into account both the distance and angle.

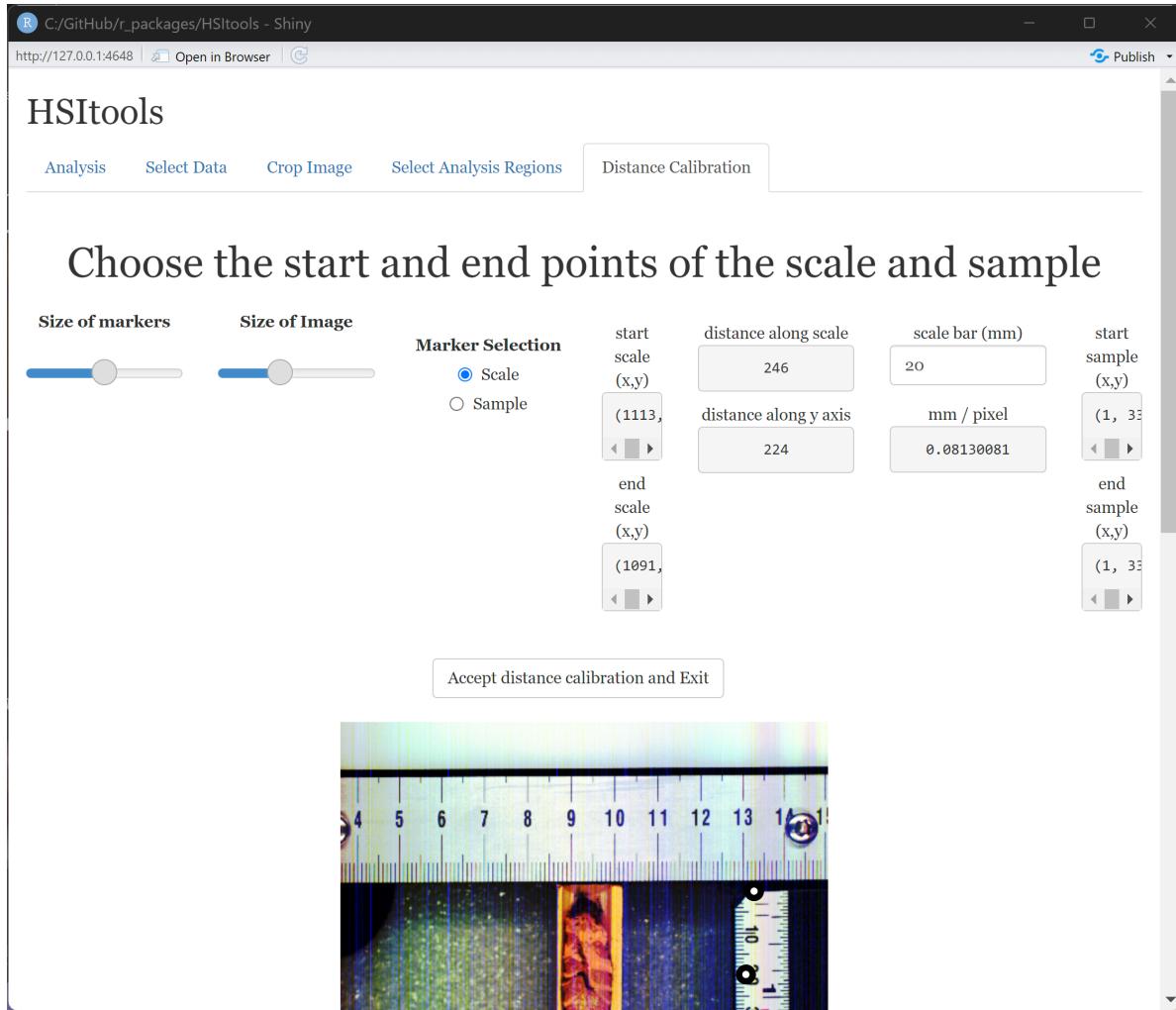


Figure 2.9: Shiny – depth calibration.

Finally, we put in the second set of markers, which tracks the actual beginning and end of the scanned sample—the true-zero is essential here.

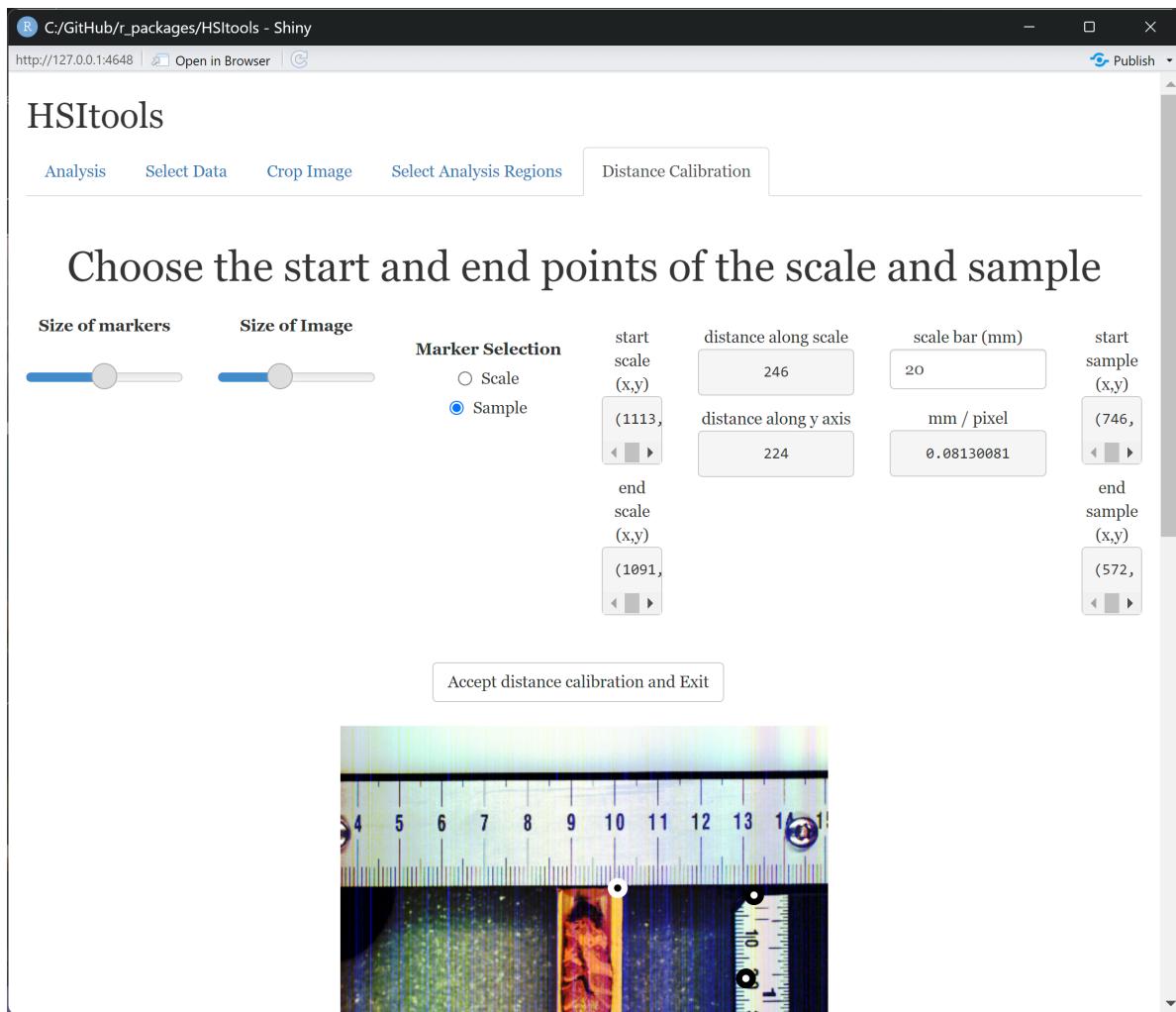


Figure 2.10: Shiny – true-zero calibration.

Once you select both sets of markers, accept and exit the app.

# 3 Preprocessing

## 3.1 Normalization

Before any spectral indices and properties are calculated, normalizing the data and expressing it as a reflectance is necessary. Here, reflectance is a fraction of the signal between the dark and white references acquired during or before the scan.

Normalization is achieved by following the equation:

$$\text{reflectance} = (d_{\text{raw}} - \text{dark}_{\text{ref}}) / (\text{white}_{\text{ref}} - \text{dark}_{\text{ref}})$$

**where:**

reflectance: normalized data cube;

$d_{\text{raw}}$ : raw captured data;

$\text{dark}_{\text{ref}}$ : dark reference data;

$\text{white}_{\text{ref}}$ : white reference data.

Which can be modified for different acquisition times for white reference:

$$\text{reflectance} = (d_{\text{raw}} - \text{dark}_{\text{ref}}) / (\text{white}_{\text{ref}} - \text{dark}_{\text{ref}}) * t_{\text{int}} \text{white} / t_{\text{int}} \text{sample}$$

**where:**

$t_{\text{int}} \text{white}$ : integration time of white reference;

$t_{\text{int}} \text{sample}$ : integration time of captured data (sample).

### 3.1.1 Normalization with Shiny output

If Shiny GUI was used for data selection, cropping, and calibration, then it would be easy to pass Shiny's output to the normalization routine. The normalized files will be written into your data's products directory.

First, this is specific only for workflow, where you'd like to calculate reflectance from the Shiny output. We must set the working directory (otherwise, we discourage this approach in R).

```
# Set working direcotory  
setwd("path/to/my_core")
```

Then, use the standard function to calculate the reflectance.

```
# Create normalized reflectance file  
reflectance <- hsi_tools_core |>  
HSItools::prepare_core()
```

- ① The Shiny GUI output.
- ② Normalization function.

**i** Multiple cores

It is planned to optionally iterate over multiple directories using, for example, the {purrr} package. This will need to get rid of setwd() call.

### 3.1.2 Normalization of the directory (no Shiny output)

If no Shiny output is available and input is not produced by hand, the normalization routine can be run without it. In such a case, the entire capture data will be normalized. However, without Shiny output, it is harder to calibrate distances properly.

Here, we normalize all the captured data with all the available layers between 400 and 1000 nm.

```
# Store the path  
path <- "data/my_core"  
  
# Create normalized reflectance file  
reflectance <- path |>  
HSItools::prepare_core(path = path, layers = c(400:1000), extent = "capture")
```

- ① Path to your core of choice, root directory. Don't go to the "capture" directory.
- ② Normalization function. The path is your core path, then the selection of layers (bands) and an explicit call informing that the entire capture extent should be normalized.

It is possible to iterate over multiple directories at once using the `{purrr}` package.

```
# Store the paths
paths <- list(
  core_1 = "data/my_core_1",
  core_2 = "data/my_core_2")①

# Or easier, with listing
paths <- fs::dir_ls("data")②

# Create normalized reflectance files
reflectance <- paths |>
  purrr::map(\(i) HSItools::prepare_core(
    path = i,
    layers = c(400:1000),
    extent = "capture"))③④
```

- ① Create a list of directories by hand.
- ② Or list root directories in your higher-level directory.
- ③ Your paths.
- ④ Normalization function iterated over multiple directories.

## 3.2 Savitzky-Golay smoothing

Applying spectral smoothing, such as the Savitzky-Golay spectral filter (Savitzky and Golay 1964), prevents spurious, random peaks and troughs from significantly influencing the calculation results.

```
reflectance <- reflectance |>①
  # Specify the file extension
  HSItools::filter_savgol()②
```

- ① Reflectance data is either in memory or from a disk.
- ② Calculation function.

### **i** Speed

This function applies a smooth filter to every pixel. Depending on the size of your data, it will take considerable time to run.

## 3.3 Median filter

You might be interested in smoothing your data spatially to remove some of the noise in the XY domain.

```
reflectance <- reflectance |>                                (1)  
# Specify the file extension  
HSItools::filter_median()                                     (2)
```

- ① Reflectance data is either in memory or from a disk.
- ② Calculation function. You can set the window (defaults to 3 by 3).

### **i** Speed

This function applies a focal median filter to every pixel. Depending on the size of your data, it will take considerable time to run. More than the Savitzky-Golay. Consider running it on the calculated index rasters and ROIs.

## 3.4 Continuum removal

We can process our data further by removing the continuum, which follows the rule of dividing the spectrum by its bounding box. This way, the spectrum becomes flatter.

```
reflectance_cr <- reflectance |>                                (1)  
HSItools::remove_continuum()                                     (2)
```

- ① Reflectance data is either in memory or from a disk.
- ② Calculation function.

# 4 Regions of interest

Selecting regions of interest (ROI) is a crucial task that requires deciding which parts of the spatial dataset will be analyzed. With HSItools, we follow two paths to select the data. The short one is based solely on the Shiny output and is limited, and the longer one is based on the Geographic Information System (GIS Software). In the spirit of the free and open source, we suggest [QGIS](#), but other apps, such as [ArcGIS Pro](#), allow the same functionality. What's important is that using GIS will enable you to freely draw shapes for data masking, a technique that helps you to focus on specific areas of the dataset while excluding others – more on this later.

## 4.1 Short path – The Shiny Path

If you opt for the Shiny path, you will work with ROIs selected in the Shiny app. These ROIs are strictly rectangular, and you don't have an immediate option to mask the data. However, you can still do it later, as you can mask any raster data. The selected regions are available within the `HSItools_core.rds` file or stored as an in-memory object.

## 4.2 Long path – The GIS Path

Here, we assume we are at least somewhat familiar with the GIS environment.

### QGIS project

It is a good idea to save the QGIS project file so you can keep the placement and customization of your layers. We suggest keeping the project name the same as your captured data.

For speed, we load the full-resolution RGB preview generated previously with the `stretch_raster_full()` function, preferably a .tiff file. Simply dragging and dropping into the empty window is enough. In the options of your new raster layer, you can increase your data's brightness or contrast, making drawing easier.

Once you're happy with your project and data presentation, start drawing.

# 5 Indices

HSItools offers a way to calculate a few of the most common indices classes used in paleoenvironmental investigations.

## 5.1 Mean reflectance (Rmean)

The most straightforward index is the mean of the reflectance values of all selected wavelengths within a given pixel. It reflects overall changes in the darkness or brightness of the captured specimen.

```
rmean <- reflectance |>  
  HSItools::calculate_rmean()  
①  
②
```

- ① Reflectance data is either in memory or from a disk.
- ② Calculation function.

## 5.2 Relative Absorption Band Depth (RABD)

A common index which

The RABD calculation has variations, but the results are generally not drastically different. You can use predefined values or provide them manually.

The output's name informs you about the calculated proxy and additional modifications to the reflectance file; here, we calculated it with Savitzky-Golay smoothed reflectance. Let's calculate one of the most common indices to estimate the total chloropigments-*a*: RABD<sub>660:670</sub>.

### 5.2.1 Variant 1 – “max”

In this variant, a minimum reflectance is found in the trough for each pixel and flexibly used for calculations.

```

rabd_max <- reflectance |>
HSItools::calculate_rabd(
  edges = proxies$rabd_b660b670$edges,
  trough = c(660:670),
  rabd_name = proxies$rabd_b660b670$proxy_name,
  rabd_type = "max")

```

- ① Reflectance data is either in memory or from a disk.
- ② Calculation function.
- ③ Edges of the trough, the broader scope.
- ④ Trough of interest, a narrower scope to find the reflectance value.
- ⑤ Name of the index to be stored in the raster data.
- ⑥ Type of the RABD calculation.

### 5.2.2 Variant 2 – “strict”

This classic variant supplies a specific wavelength to calculate RABD for every pixel. Therefore, it is RABD<sub>665</sub>.

```

rabd_strict <- reflectance |>
HSItools::calculate_rabd(
  edges = proxies$rabd_b660b670$edges,
  trough = 665,
  rabd_name = proxies$rabd_b660b670$proxy_name,
  rabd_type = "strict")

```

### 5.2.3 Variant 3 – “midpoint”

This is variant 2 (strict), with the added shortcut of always finding the middle point between the through edges—a convenience shortcut for some. Here, it equals RABD<sub>665</sub>.

```

rabd_midpoint <- reflectance |>
HSItools::calculate_rabd(
  edges = proxies$rabd_b660b670$edges,
  trough = c(660:670),
  rabd_name = proxies$rabd_b660b670$proxy_name,
  rabd_type = "mid")

```

## 5.3 Relative Absorption Band Area - ToDo

### ⚠ ToDo

This is a feature yet to be added.

```
raba <- reflectance |>
HSItools::calculate_raba(
  edges = proxies$rabd_b660b670$edges,
  trough = c(660:670),
  rabd_name = proxies$rabd_b660b670$proxy_name,
  rabd_type = "mid")
```

(1)  
(2)  
(3)  
(4)  
(5)  
(6)

- (1) Reflectance data is either in memory or from a disk.
- (2) Calculation function.
- (3) Edges of the trough, the broader scope.
- (4) Trough of interest, a narrower scope to find the reflectance value.
- (5) Name of the index to be stored in the raster data.
- (6) Type of the RABD calculation.

## 5.4 Spectral ratios

Another popular and straightforward index is band ratios, where reflectance at wavelength X is divided by reflectance at wavelength Y.

```
ratio_570630 <- reflectance |>
HSItools::calculate_band_ratio(
  edges = proxies$ratio_b570b630$edges,
  ratio_name = proxies$ratio_b570b630$proxy_name)
```

(1)  
(2)  
(3)  
(4)

- (1) Reflectance data is either in memory or from a disk.
- (2) Calculation function.
- (3) Wavelength X and Y, numerator and denominator.
- (4) Name of the index to be stored in the raster data.

## 5.5 Derivatives - ToDo

### ⚠ ToDo

This is a feature yet to be added.

Another popular and straightforward index is band ratios, where reflectance at wavelength X is divided by reflectance at wavelength Y.

```
derivative_550 <- reflectance |>  
HSItools::calculate_derivative(  
  edges = proxies$ratio_b570b630$edges,  
  ratio_name = proxies$ratio_b570b630$proxy_name)
```

- ① Reflectance data is either in memory or from a disk.
- ② Calculation function.
- ③ Wavelength of choice.
- ④ Name of the index to be stored in the raster data.

## 5.6 Differences

```
difference_650675 <- reflectance |>  
HSItools::calculate_band_difference(  
  difference_name = proxies$diff_b650b675$proxy_name,  
  edges = proxies$diff_b650b675$edges)
```

- ① Reflectance data is either in memory or from a disk.
- ② Calculation function.
- ③ Wavelength X and Y.
- ④ Name of the index to be stored in the raster data.

## 5.7 Other

### 5.7.1 Red Edge Minimum Point (REMP) - ToDo



ToDo

This is a feature yet to be added.

This index was introduced in the paper “Ghanbari, H., Zilkey, D.R., Gregory-Eaves, I., Antoniades, D., 2023. A new index for the rapid generation of chlorophyll time series from hyperspectral imaging of sediment cores. Limnology and Oceanography: Methods 21, 703–717.

```
remp <- reflectance |>                                ①  
  HSItools::calculate_remp()                            ②
```

- ① Reflectance data is either in memory or from a disk.
- ② Calculation function.

# **6 Visualizing the data**

**6.1 RGB preview**

**6.2 Index map**

**6.3 Index plot**

**6.4 Spectrum plot**

## **7 Extracting the data**

## **8 Summary**

In summary, this book has just begun.

## **References**