HSItools documentation

Maurycy Żarczyński

Table of contents

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: >= 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): >= 4.0 [link]

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")
```

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 you need to normalize data or not. 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.

2.2 Screen 2: Data choice

2.3 Screen 3: Cropping

2.4 Screen 4: ROI selection

2.5 Screen 5: Calibration

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:

Which can be modified for different acquisition setups for references:

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.

```
# Create normalized reflectance file
reflectance <- hsi_tools_core |>                                    ①
  HSItools::prepare_core()                                          ②
```

① The Shiny GUI output.
② Normalization function.

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

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.

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

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

3.2 Savitzky-Golay smoothing

Applying spectral smoothing, such as the Savitzky-Golay spectral filter (Savitzky and Golay 1964), is a good idea. This way, spurious, random peaks and throughs do not significantly influence 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.

3.3 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 |>                                    ①
  HSItools::calculate_rmean()                              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*: $\text{RABD}_{660:670}$.

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 |>                                                    ①
  HSItools::calculate_rabd(                                                  ②
    edges = proxies$rabd_b660b670$edges,                                     ③
    trough = c(660:670),                                                     ④
    rabd_name = proxies$rabd_b660b670$proxy_name,                            ⑤
    rabd_type = "max")                                                       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.

5.2.2 Variant 2 – "strict"

This classic variant supplies a specific wavelength to calculate RABD for every pixel. Therefore, it is RABD_{665}.

```
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_{665}.

```
rabd_midpoint <- reflectance |>                                              ①
  HSItools::calculate_rabd(                                                  ②
    edges = proxies$rabd_b660b670$edges,                                     ③
    trough = c(660:670),                                                     ④
    rabd_name = proxies$rabd_b660b670$proxy_name,                            ⑤
    rabd_type = "mid")                                                       ⑥
```

5.3 Relative Absorption Band Area - ToDo

```
raba <- reflectance |>                                                    ①
  HSItools::calculate_raba(                                               ②
    edges = proxies$rabd_b660b670$edges,                                  ③
    trough = c(660:670),                                                  ④
    rabd_name = proxies$rabd_b660b670$proxy_name,                         ⑤
    rabd_type = "mid")                                                    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 |>                                            ①
  HSItools::calculate_band_ratio(                                         ②
    edges = proxies$ratio_b570b630$edges,                                 ③
    ratio_name = proxies$ratio_b570b630$proxy_name)                       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

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

```
derivative_550 <- reflectance |>                                    ①
  HSItools::calculate_derivative(                                   ②
    edges = proxies$ratio_b570b630$edges,                           ③
    ratio_name = proxies$ratio_b570b630$proxy_name)                 ④
```

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 |>                                 ①
  HSItools::calculate_band_difference(                              ②
    difference_name = proxies$diff_b650b675$proxy_name,             ③
    edges = proxies$diff_b650b675$edges)                            ④
```

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

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

7 Extracting the data

8 Summary

In summary, this book has just begun.

References