BES 2025 Remote Sensing for Ecologists

Acquisition and Analysis of Field Data

2025-12-15

BRITISH
ECOLOGICAL
SOCIETY

FIELD SPECTROSCOPY
FACILITY
Natural Environment Research Council

Introduction

This worksheet will guide you through the four tutorials prepared that utilize the data you have gathered in the first session.For each tutorial, open the correct .R file found in the **BES 2025 Workshop Computer Practical Session.Rproj** folder, and work through the questions presented. This worksheet will give further information on the tasks.

Tutorial One - Using the NERC FSF RUFFS Package

The NERC FSF RUFFS (**R** s**U**ite **F**or **F**ield **S**pectroscopy) package is a comprehensive suite of tools designed to read and process spectral files acquired using common field spectrometers, namely, the Analytical Spectral Devices, Inc., Spectra Vista Corporation, and Spectral Evolution model ranges.

The functions can be classed into two groups, the `reader` functions, and the `operator` functions. The `reader` functions handle reading in .asd, .sig, and .sed files – respectively, Analytical Spectral Devices, Inc., Spectra Vista Corporation, and Spectral Evolution field spectrometer output files. The `operator` functions conduct neccessary corrections to this data to provide an interpolated, detector overlap corrected tibble dataframe for further processing in the TidyVerse R ecosystem.

Reading Spectral Files

Importing the libraries

First, import the RUFFS and TidyVerse library:

```r
library(RUFFS)
library(tidyverse)
```

Reading files

RUFFS includes sample spectral data files for demonstration. We can locate them using `system.file()`:

```r
extdata_path <- system.file("extdata", package = "RUFFS")
list.files(extdata_path)
```

```
 [1] "2_1_A_D.0000.sig"    "2_1_A_V.0000.sig"    "3_1_A_D.0000.sig"
 [4] "3_1_A_V.0000.sig"    "a_0001.sed"          "a_0002.sed"
 [7] "a_0003.sed"          "b_0001.sed"          "b_0002.sed"
[10] "b_0003.sed"          "c_0001.sed"          "c_0002.sed"
[13] "c_0003.sed"          "calcite_00000.asd"   "calcite_00001.asd"
[16] "calcite_00002.asd"   "magnesite_00000.asd" "magnesite_00001.asd"
[19] "magnesite_00002.asd" "SRT70_20240823.csv"  "stibnite_00000.asd"
[22] "stibnite_00001.asd"  "stibnite_00002.asd"
```

The `reader` function can read .asd, .sed, and .sig files. It calls upon the sub-functions `asdreader`, `svcreader`, and `psrreader` based on the argument provided to `instrument_type`. Specifying `"asd"` will read .asd files; `"svc"` will read .sig files; and `"psr"` will read .sed files.

The reader function also allows you to specify what spectral data you would like to extract, using the `data_type` argument. The options are `"reflectance"` for reflectance data from 0 to 1; `"radiance"` for radiance data (units dependent on model); and `"raw_dn"` for the digital counts. For all the examples in this vignette, we will use reflectance.

To read in .asd data:

```r
asd_files <- list.files(extdata_path, pattern = "\\.asd$",
                        full.names = TRUE)
asd_data <- reader(asd_files, data_type = "reflectance")
```

```
Auto-detected instrument type: ASD
```

```
asd_data
```

```
RUFFS Spectral Data Collection
===============================
Number of spectra: 9

Summary of spectra:
  calcite_00000.asd: 2151 bands (350.0-2500.0 nm) [ASD]
  calcite_00001.asd: 2151 bands (350.0-2500.0 nm) [ASD]
  calcite_00002.asd: 2151 bands (350.0-2500.0 nm) [ASD]
  magnesite_00000.asd: 2151 bands (350.0-2500.0 nm) [ASD]
  magnesite_00001.asd: 2151 bands (350.0-2500.0 nm) [ASD]
  magnesite_00002.asd: 2151 bands (350.0-2500.0 nm) [ASD]
  stibnite_00000.asd: 2151 bands (350.0-2500.0 nm) [ASD]
  stibnite_00001.asd: 2151 bands (350.0-2500.0 nm) [ASD]
  stibnite_00002.asd: 2151 bands (350.0-2500.0 nm) [ASD]
```

To read in .sig data:

```
svc_files <- list.files(extdata_path, pattern = "\\.sig$",
                        full.names = TRUE)
svc_data <- reader(svc_files, data_type = "reflectance")
```

```
Auto-detected instrument type: SVC
```

```
svc_data
```

```
RUFFS Spectral Data Collection
===============================
Number of spectra: 4

Summary of spectra:
  2_1_A_D.0000.sig: 1024 bands (339.7-2513.2 nm) [SVC]
  2_1_A_V.0000.sig: 1024 bands (339.7-2513.2 nm) [SVC]
  3_1_A_D.0000.sig: 1024 bands (339.7-2513.2 nm) [SVC]
  3_1_A_V.0000.sig: 1024 bands (339.7-2513.2 nm) [SVC]
```

And finally, to read in .sed data:

```
psr_files <- list.files(extdata_path, pattern = "\\.sed$",
                        full.names = TRUE)
psr_data <- reader(psr_files, data_type = "reflectance")
```

```
Auto-detected instrument type: PSR
```

```
psr_data
```

```
RUFFS Spectral Data Collection
===============================
Number of spectra: 9

Summary of spectra:
  a_0001.sed: 2151 bands (350.0-2500.0 nm) [PSR]
  a_0002.sed: 2151 bands (350.0-2500.0 nm) [PSR]
  a_0003.sed: 2151 bands (350.0-2500.0 nm) [PSR]
  b_0001.sed: 2151 bands (350.0-2500.0 nm) [PSR]
  b_0002.sed: 2151 bands (350.0-2500.0 nm) [PSR]
  b_0003.sed: 2151 bands (350.0-2500.0 nm) [PSR]
  c_0001.sed: 2151 bands (350.0-2500.0 nm) [PSR]
  c_0002.sed: 2151 bands (350.0-2500.0 nm) [PSR]
  c_0003.sed: 2151 bands (350.0-2500.0 nm) [PSR]
```

All three commands will return a `ruffs_spectra_collection`, a unique collection object which allows us to handle the metadata and data of each file.

Processing Pipeline

Overlap Removal and Interpolation

Usually, unless otherwise specified, a field spectrometer will output its wavelength values in a 1 nm scale. We can inspect the wavelengths of the first spectrum in each collection:

```
# Get the name of the first ASD spectrum
asd_names <- names(asd_data)
head(asd_data[[asd_names[1]]]$wavelength, 20)
```

```
 [1] 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368
[20] 369
```

```r
# Get the name of the first PSR spectrum
psr_names <- names(psr_data)
head(psr_data[[psr_names[1]]]$wavelength, 20)
```

```
 [1] 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368
[20] 369
```

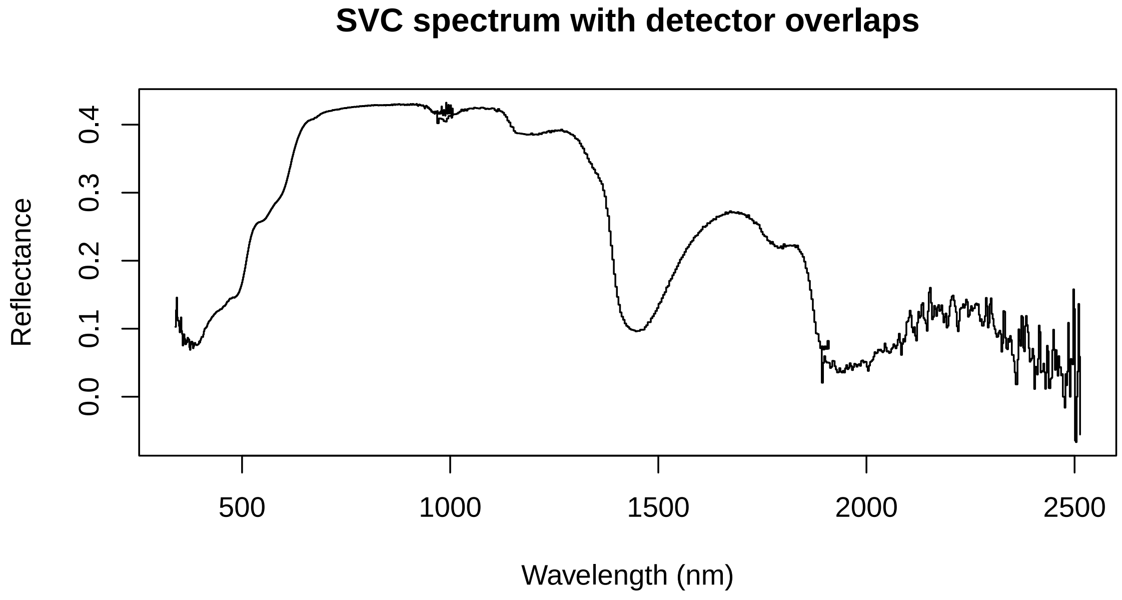
Sometimes, however, it is preferable to leave the wavelength in its default scaling, i.e. the default spectral interval between channels of each of three detectors:

```r
# Get the name of the first SVC spectrum
svc_names <- names(svc_data)
head(svc_data[[svc_names[1]]]$wavelength, 20)
```

```
 [1] 339.7 341.2 342.7 344.1 345.6 347.0 348.5 350.0 351.4 352.9 354.3 355.8
[13] 357.2 358.7 360.2 361.6 363.1 364.5 366.0 367.4
```

As a result, the different detector wavelength scales will overlap, leading to repeated wavelength data. To illustrate further, inspect the plot at the 1000 nm and 1800 nm wavelength values:

```r
# Plot the first SVC spectrum
first_svc <- svc_data[[svc_names[1]]]
plot(first_svc$wavelength, first_svc$reflectance, type = "s",
     xlab = "Wavelength (nm)", ylab = "Reflectance",
     main = "SVC spectrum with detector overlaps")
```



By leaving in a default spectral interval, one benefit is that we can choose the interpolation method, and to what spacing. It also allows us to develop unique overlap "stitching" methods.

First, we need to remove the overlap in the data using `overlap_stitching`:

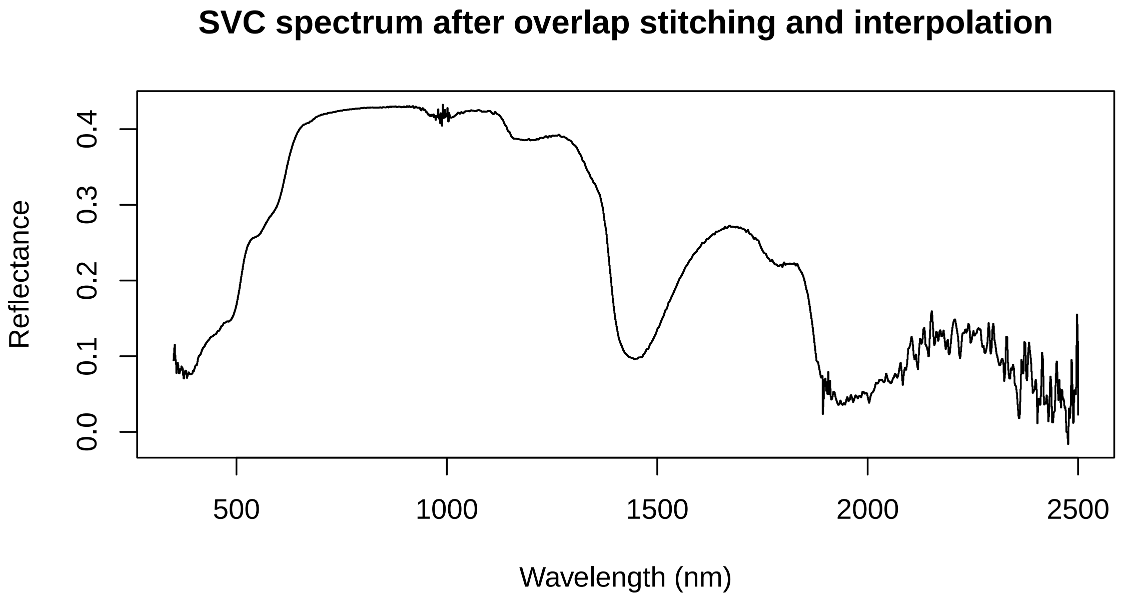
```
svc_data <- apply_to_collection(svc_data, overlap_stitching,
                                method = "average")
```

Then, to interpolate the `svc_data` to 1 nm using a linear interpolation method:

```
svc_data <- apply_to_collection(svc_data, interpolation,
                                seq(350, 2500, 1))
```

Look again at the plot of the newly interpolated, and overlap removed, SVC data:

```
# Plot the first SVC spectrum after processing
first_svc_processed <- svc_data[[svc_names[1]]]
plot(first_svc_processed$wavelength,
     first_svc_processed$reflectance,
     type = "s",
     xlab = "Wavelength (nm)", ylab = "Reflectance",
     main = "SVC spectrum after overlap stitching and interpolation")
```



To note – these processes only need to be conducted on data that has not had its overlap removed or had interpolation conducted. By default, ASD field spectrometer and Spectral Evolution field spectrometer data is interpolated with overlaps removed. However, be cautious, as this can be switched off.

Jump Correction

Regardless of whether our data needs to be interpolated and overlap stitched or not, small artefacts will remain in the data at the regions where the detector joins occurred. These are known as 'jumps', and can be removed in RUFFS using the `jump_correct` function:

```
svc_data <- apply_to_collection(svc_data, jump_correct,
                                splices = c(990, 1810),
                                reference = 2)
```

```
asd_data <- apply_to_collection(asd_data, jump_correct,
                                splices = c(1000, 1800),
                                reference = 2)
```

```
psr_data <- apply_to_collection(psr_data, jump_correct,
                                splices = c(1000, 1800),
                                reference = 2)
```

Here, splices takes the regions where the detectors join (given in the metadata, but will also be provided by the FSF if loaning a field spectrometer), and reference takes the detector type (1 is for a two detector field spectrometer; 2, which is the default, is for a three detector field spectrometer).

Grouping and Statistical Treatment

Note that our ruffs_spectra_collection files are lists of individual spectra. We can see the names of our spectra:

```
names(asd_data)
```

```
[1] "calcite_00000.asd"   "calcite_00001.asd"   "calcite_00002.asd"
[4] "magnesite_00000.asd" "magnesite_00001.asd" "magnesite_00002.asd"
[7] "stibnite_00000.asd"  "stibnite_00001.asd"  "stibnite_00002.asd"
```

We may be interested in grouping the spectra within our ruffs_spectra_collection, e.g. all instances of calcite spectra to be grouped into a calcite group. This would make it easier for subsequent statistical analysis.

To group spectra within the collection:

```
asd_data <- group_spectra(asd_data,
                          separator = "_",
                          position = 1,
                          ignore_extension = TRUE)
```

```
Grouped 9 spectra into 3 group(s):

  - 'calcite': 3 spectrum/spectra

  - 'magnesite': 3 spectrum/spectra

  - 'stibnite': 3 spectrum/spectra
```

Here, separator takes the character to group the spectra by (in this case, underscore), and position asks at which separator character should the spectra be grouped (here, 1, the first underscore).

If we were to group the SVC data by the second underscore (e.g., to group 2_1_A_D and 2_1_B_D together as 2_1), we would amend the position argument to 2:

```r
svc_data <- group_spectra(svc_data,
                          separator = "_",
                          position = 2,
                          ignore_extension = TRUE)
```

```
Grouped 4 spectra into 2 group(s):

  - '2_1': 2 spectrum/spectra

  - '3_1': 2 spectrum/spectra
```

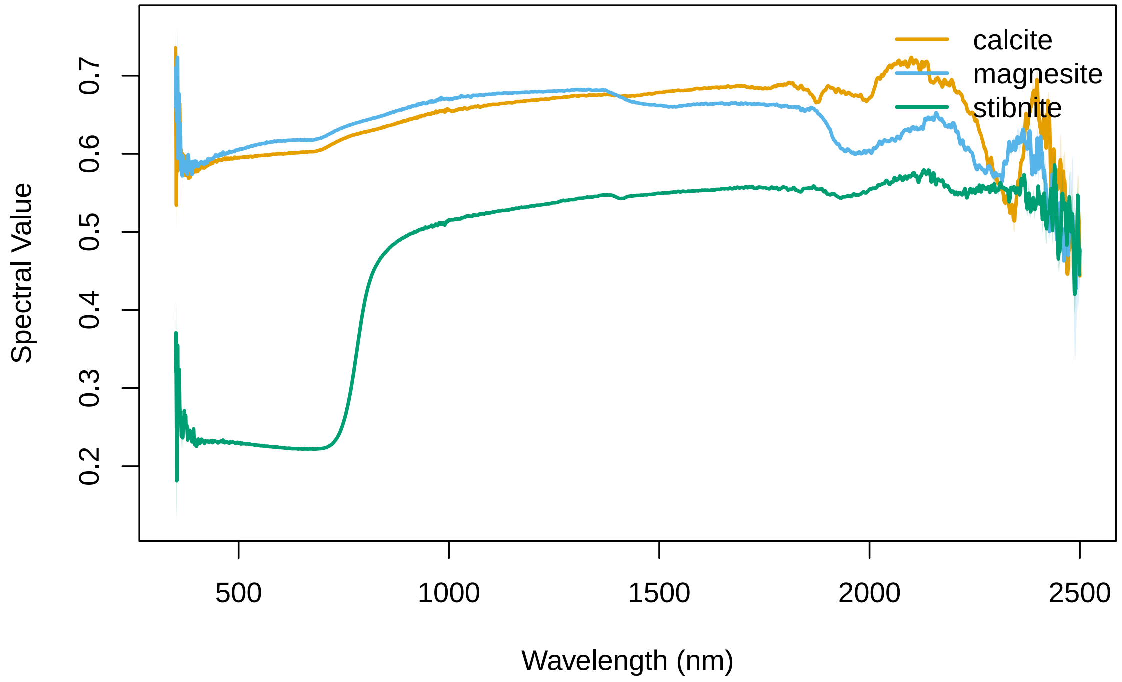
With our spectra now divided into groups, we can now conduct statistical analysis within group types. The `summarise_spectra` function will give the mean reflectance, the standard deviation within the reflectance, and the standard error within the reflectance, for each group:

```r
asd_data_summary <- summarise_spectra(asd_data)
```

```
Computed statistics for 3 group(s): mean, sd, se, n
```

Which we can now plot using the `plot_summary` function:

```r
plot_summary(asd_data_summary)
```



Exporting from `ruffs_spectra_collection` to tibble

At this point, we have now used RUFFS to read in and process our data. Moving forward, the TidyVerse ecosystem within R is an appropriate means to handle our spectral data. We can export our summary data to tibbles using the `to_tibble` function:

```
asd_tidy_data <- to_tibble(asd_data_summary)
head(asd_tidy_data)
```

```
# A tibble: 6 x 6
  wavelength  mean     sd     se     n spectrum_id
       <dbl> <dbl>  <dbl>  <dbl> <int> <chr>
1        350 0.736 0.0416 0.0240     3 calcite
2        351 0.666 0.0578 0.0334     3 calcite
3        352 0.534 0.0648 0.0374     3 calcite
4        353 0.659 0.0596 0.0344     3 calcite
```

```
5        354 0.644 0.0678 0.0391     3 calcite
6        355 0.578 0.0914 0.0528     3 calcite
```

Tutorial Two - Animal Spectral Libraries with SVC Data

The basic and advanced task in this tutorial will utilize the SVC HR-1024i data that you acquired in the first practical. As a basic task, you will run though the steps conducted above, applying to the SVC data collected in the first session, with the optional advanced task delving into a process called "convolution", which resamples hyperspectral data acquired using a field spectrometer to the multispectral bands of satellite sensors.

Basic Task - Reading, Processing, Grouping, and Statistical Treatment of SVC Data

This first task will ask you to read in the SVC HR-1024i data collected in the first session, perform corrections, group, and then conduct statistical treatment of the data, followed by plotting. The script for this can be found in **Tutorial Two - Animal Spectral Libraries with SVC Basic Task.R**, in the main R Project folder. It includes an additional overview of how to correct your relative reflectance data to absolute reflectance.

Advanced Task - Convolution

If your research relates to specific multispectral imaging sensors e.g. Sentinel 2, it can be useful to resample your hyperspectral data so that it matches the bands of your specific sensor. We can do this by convolving the hyperspectral data to a multispectral sensor's "spectral response function".

For this task, we'll look at how this package works, by convolving the data for three bands of Sentinel-2 – Band 4, Band 2 and Band 3.

Band 4 of the Sentinel-2 sensor is also know as the "Red" band. The center wavelength – the wavelength at which it has the most sensitivity – is 665 nm.

The band, however, is sensitive to other wavelengths (what is termed as its 'spectral response feature'), from 646 nm to 684 nm. The spectral response feature for Band 4, and other Sentinel 2 bands, is shown below.

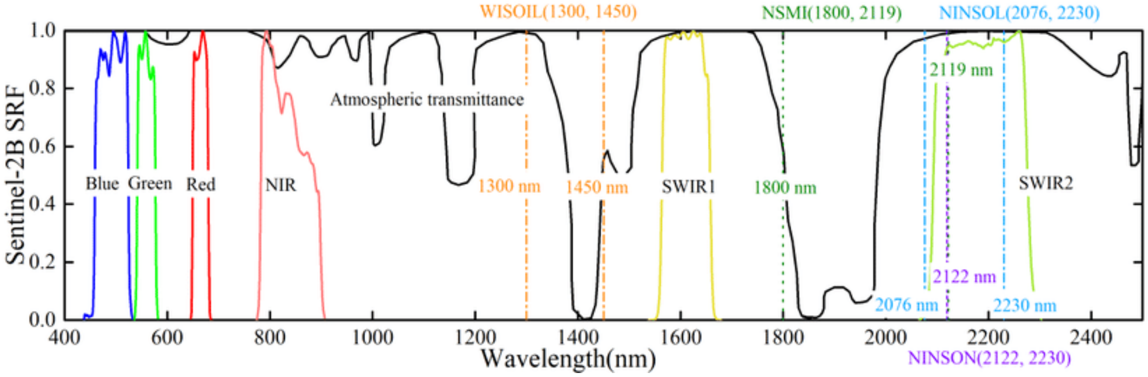


Figure 1: Sentinel-2 bands, with bandwidths

In order to accurately compare our hyperspectral data to Band 4, we must convolve the hyperspectral data within the wavelength regions of Band 4's SRF. This requires knowledge of the varying sensitvity of the band. The .csv file 'Sentinel 2 SRF.csv' contains this information.

Knowing the starting and ending wavelength of the SRF of interest, we can now conduct convolution, which mathematically can be expressed as the result of the trapezodial integration of the product of the multiplication hyperspectral data over the SRF region by the SRF, divided by the result of the trapezoidial integration of the SRF itself.

$$R_v^k = R_h \otimes S_b = \int_{v_1^k}^{v_2^k} R_h(v) S_b(v) \mathrm{d}v \Big/ \int_{v_1^k}^{v_2^k} S_b(v) \mathrm{d}v \qquad (1)$$

Open the R file in your project directory called **Tutorial Two - Animal Spectral Libraries with SVC Advanced Task.R**. You will find code that will allow you to convolve your hyperspectral data to Band 4 of Sentinel 2, using the *pracma* library. You will then be tasked with convolving your data to Band 5 and Band 6.

To help you with this task, we have provided the wavelength regions for each of the three bands you need to convolve to:

1. Band 4, "Red" - spectral range is **648 nm** to **684 nm**.
2. Band 2, "Blue" - spectral range is **439 nm** to **534 nm**
3. Band 3, "Green" - spectral range is **538 nm** to **583 nm**

Can you determine the convolved reflectance values for your groups for the red, green and blue bands of Sentinel-2?

Tutorial Three - Plant Ecology and Remote Sensing with PSR+3500 Data

Basic Task - Reading, Processing, Grouping, and Smoothing PSR Data

This first task will ask you to read in the PSR+ 3500 data collected in the first session, perform corrections, group, and then conduct statistical treatment of the data, followed by plotting. The script for this can be found in **Tutorial Three - Plant Ecology and Remote Sensing Basic Task.R**, in the main R Project folder. It includes an additional overview of how to smooth spectral data using Savitsky-Golay functions in the "signal" library.

Advanced Task - Vegetation Indices

With your data now processed and converted into absolute spectral reflectance, we can look more closely at differences between the feature types. We can use dimensionality reduction methods, such as spectral indices, to do this.

A number of spectral indices have been designed to highlight different vegetation properties. In this next section, we will use some of them to explore differences between our frature types. For a full description of the indices used, please visit:

NV5 GeoSpatial Whitepaper on Vegtation Indices

One of the most commonly used vegetation indices is the Normalized Difference Vegetation Index (NDVI). When using Sentinel-2 data, the equation to calculate NDVI is –

$$NDVI = \frac{B_8 - B_4}{B_8 + B_4} \tag{2}$$

where Band 8 is equivalent to 842 nm, and Band 4 is equivalent to 665 nm.

Open the R file in your project directory called **Tutorial Three - Plant Ecology and Remote Sensing Advanced Task.R**. You will find code that will guide you through the determination of NDVI for your PSR+3500 data set.

Use the code for NDVI determination as a framework for the following tasks:

Task – The Carotenoid Reflectance Index looks at the the concentrations of carotenoids in vegetation. Carotenoids function in light absorption processes in plants, as well as in protecting plants from the harmful effects of too much light. Higher CRI1 values mean greater carotenoid concentration relative to chlorophyll. The value of this index ranges from 0 to more than 15. The common range for green vegetation is 1 to 12:

$$CRI = \frac{1}{510} - \frac{1}{550} \tag{3}$$

Determine the Carotenoid Reflectance Index for your data set.

Task – The Plant Senescence Reflectance Index maximizes the sensitivity of the index to the ratio of bulk carotenoids (for example, alpha-carotene and beta-carotene) to chlorophyll. An increase in PSRI indicates increased canopy stress (carotenoid pigment), the onset of canopy senescence, and plant fruit ripening. Applications include vegetation health monitoring, plant physiological stress detection, and crop production and yield analysis. The value of this index ranges from -1 to 1. The common range for green vegetation is -0.1 to 0.2:

$$PSRI = \frac{680 - 500}{750} \tag{4}$$

Determine the Plant Senescence Reflectance Index for your data set.

Task – The Cellulose Absorption Index is a reflectance measurement that quantifie dried plant material. Strong absorption features present in the 2000nm to 2200nm range indicate strong presence of cellulose.

$$CAI = 0.5(2000 + 2200) - 2100 \tag{5}$$

The value range of this index ranges from -3 to more than 4. The common range for green vegetation is -2 to 4. Determine the Cellulose Absorption Index for your data set.

Tutorial Four - Landscape Spectral Libraries Basic Task with ASD FieldSpec Data

Basic Task - Reading, Processing, Grouping, and Statistics for ASD Data

This final basic task asks you, without input, to read in the ASD data found in the Tutorial Four data directory, process appropriately, conduct statistical analysis of, group, and then export to a tibble. The task is found in **Tutorial Four - Landscape Spectral Libraries Basic Task.R**. An additional task, on finding the NDVI of the individual group members, is also included. If the four groups are soil, gravel, sand and heather, what would you expect to see as the highest NDVI value?

Advanced Task - Introduction to Principal Component Analysis for Hyperspectral Data in R

This task, found in **Tutorial Four - Landscape Spectral Libraries Advanced Task.R**, guides you through the use of principal component analysis for hyperspectral data using R.

Recall from the previous tutorial that we used vegetation indices to extract information on things such as the health, senescence, and pigment presence of our plant samples. This took reflectances at only a few wavelengths out of 2,150 wavelengths to find out information about the sample. This is an example of **dimensionality reduction**.

What if we had groups which weren't neccessarily well explained by these indices? What if other wavelengths were important in discriminating against groups? If we have hyperspectral data, as we do with field spectroscopy data, we can use **Principal Component Analysis** to detect which wavelengths within our sample set are responsible for the most variation between sample groups.

The principle behind PCA involves constructing a covariance matrix from the spectral data, and decomposing it to find the directions of maximum variance. Each principal component is a linear combination of the original wavelengths, with the first component (PC1) capturing the greatest amount of variance in the dataset, the second (PC2) capturing the second-greatest variance while being orthogonal to PC1, and so on. Usually, the first three PCs can explain over 95% of the variance between spectral groups. The scores represent where each individual spectrum falls along these new component axes.

If we visualise these scores, each point in the scatter plot represents one field measurement positioned in PC space, where one dimension is PC1, and the second is PC2. The loadings indicate how much each original wavelength contributes to each principal component; high absolute loading values reveal which parts of the spectrum are most important for distinguishing between groups. For instance, if PC1 has strong positive loadings around 550 nm (green reflectance) and strong negative loadings around 1450 nm (water absorption), this component is essentially contrasting visible reflectance against water content—a pattern that might separate dry soil from vegetated surfaces.

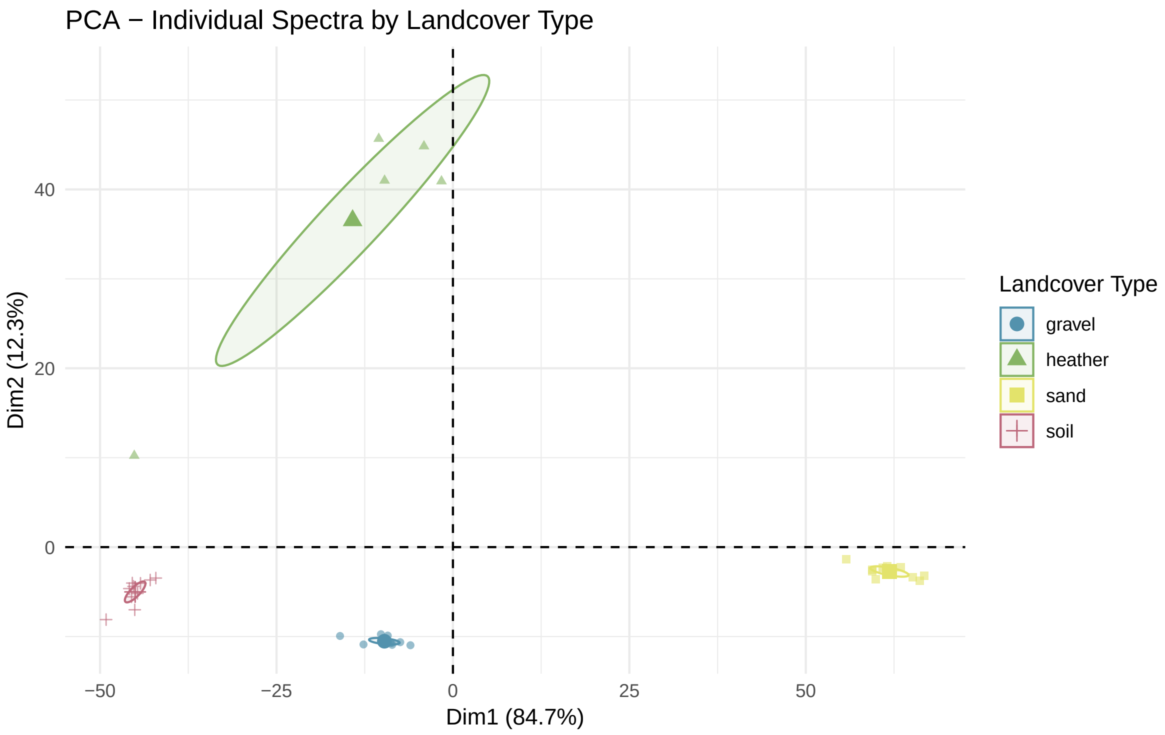


Figure 2: Scatter plot of landcover types against PC1 and PC2 of PCA

The scatter plots of PC scores reveal how well different surface types or species separate in spectral space: tightly clustered groups indicate consistent spectral signatures within a class, while well-separated clusters suggest distinct spectral properties between classes. The confidence ellipses add statistical rigor by showing whether group differences exceed natural within-group variation.

Taken together, PCA is a powerful tool for reducing large hyperspectral datasets to just a few wavelengths that can explain the variation within your samples. For vegetation health monitoring, deviations from the normal PCA space for a given species can indicate stress, disease, or phenological changes before they become visually apparent. In precision agriculture, PCA of crop spectra can distinguish nutrient deficiencies, water stress, or pest damage at early stages. The technique also translates to metabolomics and biochemical ecology, where spectroscopy methods like near-infrared spectroscopy (NIRS) use PCA to predict leaf nitrogen content, lignin concentration, or secondary metabolite profiles from spectral data.

The tools provided in **Tutorial Four - Landscape Spectral Libraries Advanced Task.R** should provide a basis for you to utilize for further PCA on other spectral data sets, or indeed, any dataset which requires dimensionality reducation.