Using hyperspectral remote sensing data to predict biodiversity

Jesús N. Pinto-Ledezma and Jeannine Cavender-Bares

In this lab we will explore some aspects of **hyperspectral** remote sensing obtained from the National Ecological Observatory Network (NEON) with the goal of predicting biodiversity from the sky. Specifically, we will use hyperspectral remote sensing data from the **NEON** Airborne Observation Platform (**AOP**) (more information here and here). As you will see, working with hyperspectral information/data is similar to work with any other information/data (e.g., species abundance, presence-absence) and consequently it can be used to calculate any metric of biodiversity. In this sense, **spectral diversity** can be considered as a dimension of biodiversity.

Note. Part of the text used in this tutorial was extracted from here with some modifications.

Set up your data and your working directory

Set up a working directory and store the data files in that directory. Tell R that this is the directory you will be using, and read in your data:

```
setwd("path/for/your/directory")
```

Install and load the following packages. When installing the package {**rhdf5**} R will ask you if you want to **"Update all/some/none? [a/s/n]:"** please in your console type **n**.

```
packages <- c("maptools", "rgdal", "raster", "neonUtilities", "rasterdiv",
              "BiocManager", "dplyr", "tidyr", "ggplot2", "plyr", "reshape2")

# Install packages not yet installed
installed_packages <- packages %in% rownames(installed.packages())

if (any(installed_packages == FALSE)) {
  install.packages(packages[!installed_packages], dependencies = TRUE)
}

BiocManager::install("rhdf5")
```

Call or load all packages

```
sapply(packages, require, character.only = TRUE)
library(rhdf5)
```

Data preparation

```
dir.create("Data/NEON")
```

The first step is to prepare the information required to download the hyperspectral data from NEON-AOP. To do that we will first download the **Terrestrial Observation System Sampling Locations** dataset. You can download it from NEON Spatial Data & Maps, specifically under the **Terrestrial Observation**

System Sampling Locations tab or by clicking **HERE**. Once you have downloaded the data, please store it within the folder **NEON** that is located within the folder **Data**

```r
NEON_plots <- readOGR(dsn = "Data/NEON/All_NEON_TOS_Plots_v8",
                      layer = "All_NEON_TOS_Plot_Polygons_V8")
```

From the spatial distribution of the NEON **sites/plots** we loaded into R, we will select the NEON site Harvard Forest & Quabbin Watershed (**HARV**).

```r
HARV_plots <- NEON_plots@data %>%
  filter(siteID == "HARV" & plotType == "distributed" & subtype == "basePlot") %>%
  arrange(plotID)

View(HARV_plots)
```

Now from the NEON site HARV we will select the plot number one (plotID = **HARV_001**) and extract the coordinates in UTM (Universal Transverse Mercator) system as spatial reference to download the hyperspectral data from the NEON-AOP.

```r
east <- HARV_plots$easting
names(east) <- HARV_plots$plotID

north <- HARV_plots$northing
names(north) <- HARV_plots$plotID

east
north

coords_HARV_001 <- c(east[1], north[1])
coords_HARV_001
```

Download hyperspectral imagery from NEON-AOP

Now we have the UTM coordinates from the plot **HARV_001** we can download the Hyperspectral Remote Sensing Data in HDF5 Format for that site.

```r
byTileAOP(dpID = "DP3.30006.001", # NEON-AOP product
          site = "HARV", # Site code
          year = "2018", # Year
          check.size = TRUE,
          easting = coords_HARV_001[1], northing = coords_HARV_001[2], # Coordinates UTM
          savepath = "Data/NEON", # Path
          token = NA)
```

Hyperspectral remote sensing data exploration

We are now ready for start using hyperspectral data. Note that the downloaded data is in a **HDF5** format, this format natively compresses data stored within it (i.e., makes it smaller) and supports data slicing, in other words, you can extract only the portions of the data that you need to work with rather than reading the entire dataset into memory.

The downloaded hyperspectral data is stored within a folder **DP3.30006.001** that in turn is stored within the folder scheme **Data/Neon**. Please pay attention to the full folder scheme, this scheme is composed of several folders with a file **NEON_D01_HARV_DP3_725000_4700000_reflectance.h5** in HDF5 format. In order to load and work with this data you might need to specify the full path to that file.

```
f <- "Data/NEON/DP3.30006.001/2018/FullSite/D01/2018_HARV_5/L3/Spectrometer/Reflectance/NEON_D01_HARV_DI
```

Note that you did not loaded all hyperspectral data into R but the path to the file, so using the path we can just call the metadata of the downloaded data.

HDF5 file structure

When we are exploring the data structure in a HDF5 file, we really need to pay attention to the first two columns, these two columns is informing us about the location of the data (**group**) and the name of the data (**name**) stored into the file. For example the **Map_info** dataset is located in **/HARV/Reflectance/Metadata/Coordinate_System** and the **Reflectance** dataset under **/HARV**.

The **wavelength** dataset contains the middle wavelength values for each band in the dataset and the **Reflectance** dataset contains the image data, these two datasets are going to be used for both data processing and visualization.

```
View(h5ls(f, all = TRUE))
```

On bands and wavelengths

A band represents a group of wavelengths. For example, the wavelength values between 695 nm and 700 nm might be one band as captured by an imaging spectrometer. The imaging spectrometer collects reflected light energy in a pixel for light in that band. When we are working with a multispectral (e.g., Landsat, MODIS) or hyperspectral (e.g., NEON-AOP - NASA/JPL AVIRIS-NG) dataset, the band information is reported as the center wavelength value. This value represents the center point value of the wavelengths represented in that band. Thus in a band spanning 695-700 nm, the center would be 697.5 nm.

```
# Get information about the wavelengths of HARV plot 001
wlInfo <- h5readAttributes(f, "/HARV/Reflectance/Metadata/Spectral_Data/Wavelength")

wlInfo
```

```
# Read wavelengths from the HDF5 file
WL <- h5read(f, "/HARV/Reflectance/Metadata/Spectral_Data/Wavelength")

head(WL)
tail(WL)
```

```
# Extract reflectance metadata
reflInfo <- h5readAttributes(f, "/HARV/Reflectance/Reflectance_Data")

reflInfo
```

For example, the center wavelength value associated with the band 34 is **546.8372**

```
WL[34]
```

The HDF5 read function reads data in the order: **Bands**, **Cols**, **Rows**. Let's get these information from the reflectance metadata.

```
# Read dimensions of the hyperspectral data
nRows <- reflInfo$Dimensions[1]
nCols <- reflInfo$Dimensions[2]
nBands <- reflInfo$Dimensions[3]

nRows
nCols
nBands
```

Now using the the information obtained from the reflectance metadata let's extract the **band 34**.

```
# Extract or "slice" data for band 34 from the HDF5 file
b34 <- h5read(f, "/HARV/Reflectance/Reflectance_Data",
             index = list(34, 1:nCols, 1:nRows)) # get band 34

# what type of object is b34?
class(b34)

## [1] "array"
```

The returned object is an **array**, the arrays are matrices with more than 2 dimensions, i.e., are matrices stacked or piled in a single object.

```
# convert from array to matrix by selecting only the first band
b34 <- b34[1,,]

# check it
class(b34)

# plot the image
image(b34)
```

The previous image is hard to visually interpret, let's log the data and see what happens.

```
image(log(b34))
```

Data cleaning

An image data in raster format will often contain a **data ignore** value and a **scale factor**. The data ignore value represents pixels where there are no data. Usually, no data values may be attributed to the sensor not collecting data in the area of the image or to processing results which yield null values.

From the reflectance metadata we can define the ignore value as **-9999**. Thus, let's set all pixels with a value == -9999 to NA (no value).

```
# there is NO data value in our raster - let's define it
myNoDataValue <- as.numeric(reflInfo$Data_Ignore_Value)
myNoDataValue

# set all values equal to -9999 to NA
b34[b34 == myNoDataValue] <- NA

# plot the image now
image(b34)
```

```
# We need to transpose x and y values in order for our final image to plot properly
b34 <- t(b34)
image(log(b34), main = "Transposed Image")
```

Creating a georeferenced raster

In order to get a raster file suitable for further analysis, we first need to define the Coordinate reference system (CRS) of the raster. Again, we obtian the necessary onformation form the HDF5 file.

```
# Extract the EPSG from the h5 dataset
myEPSG <- h5read(f, "/HARV/Reflectance/Metadata/Coordinate_System/EPSG Code")
```

```
# convert the EPSG code to a CRS string
myCRS <- crs(paste0("+init=epsg:", myEPSG))
```

Define final raster with projection info.

```
b34ras <- raster(b34, crs = myCRS)

# view the raster attributes
b34ras
```

Let's take a look at the georeferenced raster. Take note of the coordinates on the x and y axis.

```
image(log(b34ras),
      xlab = "UTM Easting",
      ylab = "UTM Northing",
      main = "Properly Oriented Raster")
```

Next we define the extents of our raster. The extents will be used to calculate the raster's resolution. We get this information from the reflectance information obtained in a previous step.

```
# Grab the UTM coordinates of the spatial extent
xMin <- reflInfo$Spatial_Extent_meters[1]
xMax <- reflInfo$Spatial_Extent_meters[2]
yMin <- reflInfo$Spatial_Extent_meters[3]
yMax <- reflInfo$Spatial_Extent_meters[4]
```

```
# define the extent (left, right, top, bottom)
rasExt <- extent(xMin, xMax, yMin, yMax)

# view the extent to make sure that it looks right
rasExt
```

```
# assign the spatial extent to the raster
extent(b34ras) <- rasExt

# look at raster attributes
b34ras
```

```
# let's change the colors of our raster and adjust the zlims
col <- terrain.colors(25)

image(b34ras,
      xlab = "UTM Easting",
      ylab = "UTM Northing",
      main = "Raster with Custom Colors",
      col = col,
      zlim = c(0, 1000))
```

We can now save the created raster.

```
# write out the raster as a geotiff
writeRaster(b34ras,
            file = "Data/NEON/DP3.30006.001/HARV_plot_001_band_34.tif",
            format = "GTiff",
            overwrite = TRUE)
```

Creating a RGB raster

In the previous step we created a raster file of a single band, but each hyperspectral data from the NEON-AOP contains 426 bands. In this step we will construct a raster stack file, i.e., a raster with N bands, in other words, a raster of rasters. You can do it manually as we did for the band 34, but let's take advantage of **R** and write a function that do the work for us.

```r
# file: the hdf file
# band: the band you want to process
# noDataValue: values to be omitted
# extent: raster extent
# CRS: coordinates system
# returns: a matrix containing the reflectance data for the specific band

band2Raster <- function(file, band, noDataValue, extent, CRS){
    # first, read in the raster
    out <- h5read(file, "/HARV/Reflectance/Reflectance_Data",
                  index = list(band, NULL, NULL)) # path to the HDF5 file
      # Convert from array to matrix
      out <- (out[1,,]) # output
      # transpose data to fix flipped row and column order
    # depending upon how your data are formatted you might not have to perform this
    # step.
      out <- t(out)
    # assign data ignore values to NA
    # note, you might chose to assign values of 15000 to NA
    out[out == myNoDataValue] <- NA

    # turn the out object into a raster
    outr <- raster(out, crs = CRS)

    # assign the extents to the raster
    extent(outr) <- extent

    # return the raster object
    return(outr)
}
```

Now apply the function to create a RGB raster file, to do this, we will use the bands 58, 34 and 19, respectively.

```r
# create a list of the bands we want in our stack
rgb <- list(58, 34, 19)

# lapply tells R to apply the function to each element in the list
rgb_harv <- lapply(rgb, FUN = band2Raster, file = f,
                   noDataValue = myNoDataValue,
                   extent = rasExt,
                   CRS = myCRS)

# check out the properties or rgb_rast
# note that it displays properties of 3 rasters.
rgb_harv
```

Success!!! we created a list of three rasters. Now in order to get the raster stack just apply the function **stack()** from the package {raster}.

```
# Create a raster stack from our list of rasters
rgb_harv_stack <- stack(rgb_harv)
rgb_harv_stack
```

As a final step, let's assign the names of the bands to the raster stack object.

```
# Create a list of band names
bandNames <- paste("Band_", unlist(rgb), sep = "")

# set the rasterStack's names equal to the list of bandNames created above
names(rgb_harv_stack) <- bandNames

# check properties of the raster list - note the band names
rgb_harv_stack

# scale the data as specified in the reflInfo$Scale Factor
rgb_harv_stack <- rgb_harv_stack/as.integer(reflInfo$Scale_Factor)

# plot one raster in the stack to make sure things look OK.
plot(rgb_harv_stack$Band_58, main = "Band 58")
```

And plot resulting RGB raster.

```
# create a 3 band RGB image
plotRGB(rgb_harv_stack,
        r = 1, g = 2, b = 3,
        stretch = "lin")
```

Cool, right?

As we did with the band 34, let's save the RGB raster in a GeoTIFF format.

```
# write out final raster
writeRaster(rgb_harv_stack, file = "Data/NEON/DP3.30006.001/HARV_plot_001_RGB.tif",
            format = "GTiff", overwrite = TRUE)
```

Vegetation indices calculation

Now, we will calculate vegetation indices, specifically, we will calculate the Normalized Difference Vegetation Index (NDVI).

The $NDVI$ is computed as the difference between near-infrared (NIR) and red (RED) reflectance divided by their sum.:

$$NDVI = \frac{NIR - R}{NIR + R}$$

To calculate the **NDVI** from the NEON-AOP hyperspectral data, first select the bands 58 (red) and 90 (NIR) and then create a raster stack as we did for the RGB raster.

```
# Calculate NDVI
# select bands to use in calculation (red, NIR)
ndvi_bands <- c(58, 90) #bands c(58, 90) in full NEON hyperspectral dataset

# create raster list and then a stack using those two bands
ndvi_harv <- lapply(ndvi_bands, FUN = band2Raster, file = f,
                    noDataValue = myNoDataValue,
                    extent = rasExt, CRS = myCRS)

ndvi_harv <- stack(ndvi_harv)

# make the names pretty
bandNDVINames <- paste("Band_", unlist(ndvi_bands), sep = "")
names(ndvi_harv) <- bandNDVINames

# view the properties of the new raster stack
ndvi_harv
```

Write a function for **NDVI** calculation.

```
#calculate NDVI
NDVI_func <- function(ras) {
      (ras[,2] - ras[,1])/(ras[,2]+ras[,1])
}
```

Apply the function and plot the result.

```
ndvi_calc <- calc(ndvi_harv, fun = NDVI_func)

plot(ndvi_calc, main = "NDVI for the NEON HARV Field Site")
```

Now, play with breaks and colors to create a meaningful map add a color map with 4 colors.

```
myCol <- rev(terrain.colors(4)) # use the 'rev()' function to put green as the highest NDVI value
# add breaks to the colormap, including lowest and highest values (4 breaks = 3 segments)
brk <- c(0, .25, .5, .75, 1)

# plot the image using breaks
plot(ndvi_calc, main = "NDVI for the NEON HARV Field Site", col = myCol, breaks = brk)
```

We can save the resulting NDVI raster.

```
writeRaster(ndvi_calc, file = "Data/NEON/DP3.30006.001/HARV_plot_001_NDVI.tif",
            format = "GTiff", overwrite = TRUE)
```

Plot spectral signatures derived from hyperspectral remote sensing data

Now we will extract all reflectance values for a selected pixel in the hyperspectral data from HARV and use this values to plot its spectral signatures. For practice purpose we will select a pixel at the position (100, 35), in other words, we are selecting the pixel at the row 100 and column 35. In addition to get the the reflectance for all bands we need to inform an empty space (as NULL).

```
# extract all bands from a single pixel
aPixel <- h5read(f, "/HARV/Reflectance/Reflectance_Data", index = list(NULL, 100, 35))

class(aPixel)

# The line above generates a vector of reflectance values.
# Next, we reshape the data and turn them into a dataframe
b <- adply(aPixel, c(1))

class(b)
```

```
# create clean data frame
aPixeldf <- b[2]

# add wavelength data to matrix
aPixeldf$Wavelength <- WL

head(aPixeldf)
```

Now select the scale factor from the reflectance object and scale the reflectance values for all bands.

```
scaleFact <- reflInfo$Scale_Factor

# add scaled data column to the data frame
aPixeldf$scaled <- (aPixeldf$V1/as.vector(scaleFact))

# make nice column names
names(aPixeldf) <- c('Reflectance', 'Wavelength', 'ScaledReflectance')

head(aPixeldf)
tail(aPixeldf)
```

As a last step let's plot the scaled reflectance as a function of the wavelength.

```
ggplot(data = aPixeldf) +
   geom_line(aes(x = Wavelength, y = ScaledReflectance)) +
   xlab("Wavelength (nm)") +
   ylab("Reflectance")
```

Select pixels and compare spectral signatures

In the previous step we selected and arbitrary pixel at (100, 35), however when we are working with real data we might need to have the spatial position of objects on the ground (e.g., GPS points). Let's select 5 spatial points by visually inspecting the **RGB** we just created. Below is an image with five points, you can select other five or more points if you desire.

To select the five points you first need to plot the RGB raster.

```
plotRGB(rgb_harv_stack,
        r = 1, g = 2, b = 3,
        stretch = "lin")
```

Then using the function **click** from the package {raster} we select the five points.

```
# change plotting parameters to better see the points and numbers generated from clicking
par(col = "red", cex = 3)

# use the 'click' function
clk <- click(rgb_harv_stack, id = T, xy = T, cell = T,
          type = "p", pch = 16, col = "magenta", col.lab = "red")
```

Once you have clicked the five points, press the ESC key to save your clicked points and close the function before moving on to the next step. If you make a mistake in the step, run the plotRGB() function again to start over.

Convert raster pixel number into row and column that we will use to extract the spectral signatures for the five points.

```r
clk$row <- clk$cell%/%nrow(rgb_harv_stack) + 1 # add 1 because R is 1-indexed
clk$col <- clk$cell%%ncol(rgb_harv_stack)
```

Again, you can assign the spectral signature for each pixel manually (as we did for the pixel (100, 35)) but that is it not efficient by any means, instead we can write a **for loop** that do the works for us.

First, we will create a new dataframe object from the band wavelengths so that we can add the reflectance values for each cover type

```r
Pixel_df <- as.data.frame(WL)

# loop through each of the cells that we selected
for(i in 1:length(clk$cell)){
# extract Spectra from a single pixel
  aPixel <- h5read(f, "/HARV/Reflectance/Reflectance_Data",
                  index = list(NULL, clk$col[i], clk$row[i]))

# scale reflectance values from 0-1
  aPixel <- aPixel/as.vector(scaleFact)

# reshape the data and turn into dataframe
  b <- adply(aPixel, c(1))

# rename the column that we just created
  names(b)[2] <- paste0("Point_", i)

# add reflectance values for this pixel to our combined data.frame called Pixel_df
  Pixel_df <- cbind(Pixel_df, b[2])
}
```

Explore the data frame with reflectance values for the five points.

```r
head(Pixel_df)
tail(Pixel_df)
```

As a last step let's transform the **pixel__df** data frame to a format that {ggplot2} like.

```r
# Use the melt() function to reshape the dataframe into a format that ggplot prefers
Pixel.melt <- melt(Pixel_df, id.vars = "WL", value.name = "Reflectance")
```

Now, let's plot some spectral signatures.

```r
ggplot() +
  geom_line(data = Pixel.melt, mapping = aes(x = WL,
                                             y = Reflectance,
                                             color = variable), lwd = 1.5) +
  scale_colour_manual(values = c("green2", "green4", "chartreuse3", "tan4", "blue3"),
                      labels = c("Forest1", "Forest2", "Forest3", "Soil", "Water"))+
  labs(color = "Cover Type") +
  ggtitle("Land cover spectral signatures") +
  theme(plot.title = element_text(hjust = 0.5, size = 20)) +
  xlab("Wavelength")
```

Cool! we just made a plot with the spectral signatures for five points at **HARV** site, but there are some anomalies in the plot which difficult its interpretation. Those anomalies around 1400 nm and 1850 nm correspond to two major atmospheric absorbtion bands, i.e., regions in the spectra where gasses in the atmosphere (primarily carbon dioxide and water vapor) absorb radiation, and therefore, obscure the reflected radiation that the imaging spectrometer measures.

To eliminate those anomalies we first might need to select and eliminate those bands manually. Happily, the reflectance metadata contains the lower and upper bound of each of those **atmopheric absopbtion bands**. Let's read those bands and plot rectangles where the reflectance measurements are obscured by atmospheric absorbtion.

```
# grab Reflectance metadata (which contains absorption band limits)
reflMetadata <- h5readAttributes(f, "/HARV/Reflectance" )

ab1 <- reflMetadata$Band_Window_1_Nanometers
ab2 <- reflMetadata$Band_Window_2_Nanometers

ab1
ab2
```

```
# Plot spectral signatures again with rectangles showing the absorption bands
ggplot() +
  geom_line(data = Pixel.melt, mapping = aes(x = WL,
                                             y = Reflectance,
                                             color = variable), lwd = 1.5) +
  geom_rect(mapping = aes(ymin = min(Pixel.melt$Reflectance),
                          ymax = max(Pixel.melt$Reflectance),
                          xmin = ab1[1], xmax = ab1[2]),
            color = "black", fill = "grey40", alpha = 0.8) +
  geom_rect(mapping = aes(ymin = min(Pixel.melt$Reflectance),
                          ymax = max(Pixel.melt$Reflectance),
                          xmin = ab2[1], xmax = ab2[2]),
            color = "black", fill = "grey40", alpha = 0.8) +
  scale_colour_manual(values = c("green2", "green4", "chartreuse3", "tan4", "blue3"),
                      labels = c("Forest1", "Forest2", "Forest3", "Soil", "Water")) +
  labs(color = "Cover Type") +
  ggtitle("Land cover spectral signatures") +
  theme(plot.title = element_text(hjust = 0.5, size = 20)) +
  xlab("Wavelength")
```

By inspecting the plot we can confirm that the sections of the spectra with anomalies are within the atmospheric absortion bands. Using the absorption band limits we can remove the sections with anomalies and plot masked spectral signatures for the five spatial points.

```
# Duplicate the spectral signatures into a new data.frame
Pixel.melt.masked <- Pixel.melt

# Mask out all values within each of the two atmospheric absorbtion bands
Pixel.melt.masked[Pixel.melt.masked$WL >
                    ab1[1] & Pixel.melt.masked$WL < ab1[2], ]$Reflectance <- NA

Pixel.melt.masked[Pixel.melt.masked$WL >
                    ab2[1] & Pixel.melt.masked$WL < ab2[2], ]$Reflectance <- NA

head(Pixel.melt.masked)
```

Plot the masked spectral signatures

```
ggplot() +
  geom_line(data = Pixel.melt.masked, mapping = aes(x = WL,
                                                    y = Reflectance,
                                                    color = variable), lwd = 1.5) +
  scale_colour_manual(values = c("green2", "green4", "chartreuse3","tan4","blue3"),
                      labels = c("Forest1", "Forest2", "Forest3", "Soil", "Water"))+
  labs(color = "Cover Type")+
  ggtitle("Land cover spectral signatures")+
  theme(plot.title = element_text(hjust = 0.5, size=20))+
  xlab("Wavelength")
```

It's always good practice to close the H5 connection before moving on.

```
# close the H5 file
H5close()
```

Clean your R environment.

```
rm(list = ls())
```

Calculate metrics of biodiversity using remote sensing data

As a final exercise we will estimate some diversity metrics directly from the results obtained in the previous steps. Specifically we will use the **NDVI** raster we created before.

```
harv_ndvi <- raster("Data/NEON/DP3.30006.001/HARV_plot_001_NDVI.tif")

plot(harv_ndvi)
```

Now using the NDVI raster we will estimate the **Shannon** and **Hill** diversity metrics for each pixel. To do that we will use the package {**rasterdiv**} (Thouverai et al. 2021). Given the spatial extent and resolution of the NDVI raster file, this process will take some time (~7 minutes in Jesús's computer) to finish.

Shannon's diversity index (H')

The Shannon's diversity index is one of the most common metrics used to estimate diversity from remote sensing data. This index is calculated as:

$$H = -\sum_{n=1}^{N} p_i ln_b(p_i) = 1$$

Where p_i is the proportional abundance of pixel i and b is the base of the logarithm. It is most popular to use natural logarithms, but some argue for base $b = 2$. **Shannon's H** is a dimensionless metric, in other words, it consider differences in the relative abundance among pixel values, but not their relative spectral distance, i.e. the distance among spectral values (Thouverai et al. 2021).

In this example we are using a parallel computation which allow to obtain results kickly, however if you have some issues running this part, please remove the argument **np = 2** from the below example.

```
library(doParallel)
# Computes Shannon's diversity index (H') on different classes of numeric matrices using a moving windo
HARV_shannon <- Shannon(harv_ndvi, window = 5, np = 2)
```

Plot the results for Shannon's diversity.

```
plot(HARV_shannon)
```

Hill's generalized entropy diversity index

The Hill's generalized entropy diversity index is based on the effective number of species of $H\alpha$, i.e., the number of species that would lead to the diversity H if the species are equally abundant (Scheiner et al. 2017, Cavender-Bares et al. 2020). An important component in the Hill's diversity is the α component, thus, different orders of α result in different diversity measures; for example, $\alpha = 0$ is simply species richness, $\alpha = 1$ gives the exponential of Shannon's entropy index, and $\alpha = 2$ gives the inverse of Simpson's concentration index (Cavender-Bares et al. 2020).

$$H_\alpha = (\sum_{n=1}^{N} p_i^\alpha)^{\frac{1}{1-\alpha}}$$

Let's compute the Hill's diversity with an $\alpha = 1$ and compare the results with the **Shannon's** metric.

```
# Computes Hill's index of diversity (Hill numbers) on different classes of numeric matrices using a mo
HARV_hill <- Hill(harv_ndvi, alpha = 1, window = 5, np = 2, rasterOut = TRUE)
```

```
plot(HARV_hill[[1]])
```

```
cor.test(values(HARV_shannon), values(HARV_hill[[1]]))
```

As indicated before, when $\alpha = 1$ in the Hill's calculation it will resemble the exponential of Shannon's entropy index.

That's it!

The challenge

This was a very long lab and will take sometime to digest all the information. Thus the challenge for this lab is simple. Just select five additional spatial points and plot their spectral signatures.

References

Cavender-Bares, J., Schweiger, A. K., Pinto-Ledezma, J. N., & Meireles, J. E. (2020). Applying Remote Sensing to Biodiversity Science. Remote Sensing of Plant Biodiversity, 13–42. doi:10.1007/978-3-030-33157-3_2.

NEON (National Ecological Observatory Network). Spectrometer orthorectified surface directional reflectance - mosaic, RELEASE-2021 (DP3.30006.001). https://doi.org/10.48443/qeae-3x15. Dataset accessed from https://data.neonscience.org on April 24, 2021.

Scheiner, S. M., Kosman, E., Presley, S. J., & Willig, M. R. (2017). Decomposing functional diversity. Methods in Ecology and Evolution, 8(7), 809–820. doi:10.1111/2041-210x.12696.

Thouverai, E., Marcantonio, M., Bacaro, G., Re, D. D., Iannacito, M., Marchetto, E., … Rocchini, D. (2021). Measuring diversity from space: a global view of the free and open source rasterdiv R package under a coding perspective. Community Ecology, 22(1), 1–11. doi:10.1007/s42974-021-00042-x.