Biodiversity in the Arctic Tundra using hyperspectral images

Patrick Angst

2024-11-18

Introduction

This document outlines the progress of my masters thesis. The objective of this project is to assess biodiversity in the Arctic Tundra using spectral species analysis.The key tasks include:

- Data collection and pre-processing
- Calculating relative abundance and variance for each species.
- Identifying significant species based on defined thresholds.
- Use the R-package BiodivmapR package to calculate α and β- diversity

---

Data Preparation and Code

The following steps were taken to prepare the data and perform the analysis:

Define project parameters

All of the scripts access the information set in the parameter document 00_Project_Parameter.R

```
# filename of the raw hyperspectral image
file_name <- 'ang20190712t231624_rfl_v2v2_img'

# definition of the subzone (c, d or e)
subzone <- 'c'
file_name_rectified <- paste0(file_name,'_rectified')

# definition of the thresholdes for the mask creation
ndwi_threshold <- 0.1
ndvi_threshold <- 0.3
savi_threshold <- 0.2
savi_L <- 0.5

#create additional variables for the masks
mask_name_suffix <- gsub("\\.", "", savi_threshold)
mask_name <- paste0(file_name_rectified,'_savi_mask_',mask_name_suffix)

# get the base path of the project
base_path <- getwd()
```

Create an RGB image of the flightstrip

For quick analysis the RGB bands are extracted from the hyperspectal image and saved as a GeoTiff file 01_Create_RGB.R

```
# Clear workspace and graphics
rm(list = ls())
graphics.off()

# Define parameter script
source('00_Project_Parameter.R')


raw_image_file_path <- paste0(base_path,'/data/hs_raw_image/',file_name)
rgb_image_file_path <- paste0(base_path,'/data/rgb/',file_name,'_rgb.tif')

# Define the bands you want to select
bandselection <- "-b 59 -b 34 -b 20"

# GDAL translate command to extract the specified bands and create a new image
gdal_translate_command <- sprintf("gdal_translate %s -of GTiff %s %s", bandselection, raw_image_file_pa

# Execute the GDAL translate command
system(gdal_translate_command)

# Check if the output file was created successfully
if (file.exists(rgb_image_file_path)) {
  cat("Output file created successfully:", rgb_image_file_path, "\n")

  # GDAL edit command to set color interpretation for each band
  gdal_edit_command <- sprintf("gdal_edit.py -colorinterp_1 Red -colorinterp_2 Green -colorinterp_3 Blu

  # Execute the GDAL edit command
  system(gdal_edit_command)

  # Confirm that color interpretation was set successfully
  cat("Color interpretation set to RGB for each band in", rgb_image_file_path, "\n")

} else {
  cat("Failed to create output file.\n")
}
```

Image reactification

To follow the format set by the BiodivmapR package, the raw hyperspectral image has to be transformed. Using gdalwarp, the no-data values -9999 have to be replaced with 0 and the image organization has to be set to BIL (Band interleaved by line). 02_Rectify_Image.R

```
# Clear workspace and graphics
rm(list = ls())
graphics.off()

library(sf)
library(terra)
```

```r
# Define parameter script
source('00_Project_Parameter.R')

#boundary_file_path <- paste0(base_path, '/cutline/crop_large/crop_large.shp')
raw_image_file_path <- paste0(base_path, '/data/hs_raw_image/', file_name)
rectified_image_file_path <- paste0(base_path, '/data/rectified/', file_name_rectified)
rectified_hdr_file_path <- paste0(rectified_image_file_path, '.hdr')

target_srs <- "EPSG:32604"  # Define target CRS

gdal_command_rectify <- sprintf(
  "gdalwarp -of ENVI -co INTERLEAVE=BIL -srcnodata -9999 -dstnodata 0 %s %s",
  raw_image_file_path,
  rectified_image_file_path
)

# Print the command to check if it's correctly formed
print(gdal_command_rectify)

# Execute the command in R
system(gdal_command_rectify)

# ============================================================================
# Define wavelength information to add to .hdr file
wavelength_values <- c(
  376.719576 ,
  381.729576 ,
  ...
)

wavelength_units <- "Nanometers"
wavelength_line <- paste("wavelength = {", paste(wavelength_values, collapse = " , "), "}")
units_line <- paste("wavelength units =", wavelength_units)

# ============================================================================
# Append wavelength information to the .hdr file
if (file.exists(rectified_hdr_file_path)) {
  # Read the existing content of the .hdr file
  hdr_content <- readLines(rectified_hdr_file_path)

  # Append the wavelength information at the end of the file
  hdr_content <- c(hdr_content, units_line, wavelength_line)

  # Write the updated content back to the .hdr file
  writeLines(hdr_content, rectified_hdr_file_path)

  cat("Wavelength information successfully added to the .hdr file.\n")
} else {
  cat("Error: The .hdr file does not exist. Check the file path.\n")
}

print(paste0('Rectification done for ', file_name))
```

Mask creation

Since not all pixels have to be processed, only the "valid" have to be selected. For this purpose, a Soil Adjusted Vegetation Index (SAVI) mask is created. 03_Create_MASK.R

```
# clean environment
rm(list=ls(all=TRUE));gc()
graphics.off()

library(terra)

# Define parameter script
source('00_Project_Parameter.R')

file_name <- file_name_rectified
cell <- paste0(base_path,'/data/rectified/',file_name)
tile <- rast(file.path(cell))

# Plot the RGB image for a quick check
plot(ext(tile))
plotRGB(tile, add=T, r=54, g=36, b=20, stretch="lin")


# Calculate the mean of a few values of the near infrared bands (used for NDWI and SAVI)
NIR_average <- mean(tile[[c(86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103,
# Calculate green band averages (used for NDWI)
green_average <- mean(tile[[c(26, 27, 28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45)]])
# Calculate red band averages (used for SAVI)
red_average <- mean(tile[[c(56, 57, 58, 59, 60, 61, 62, 63, 64, 65)]])

# Plot histogram of the distribution of the values
hist(NIR_average, breaks = seq(terra::minmax(NIR_average)[1], terra::minmax(NIR_average)[2] + 0.05, by =
     main = "Histogram of NIR average", xlab = "NIR_average")


################################################################################
# Create NDWI Mask NDWI = (Red - NIR) / (Red + NIR)
################################################################################

# Calculate the NDWI
NDWI <- (green_average-NIR_average)/(green_average+NIR_average)

# Plot histogram of the value distribution
hist(NDWI, breaks = seq(terra::minmax(NDWI)[1], terra::minmax(NDWI)[2] + 0.05, by = 0.01),
     main = "Histogram of NDWI", xlab = "NDWI")

# Create the NDWI mask (binary values) with a threshold of 0.1
#ndwi_threshold <- ndwi_threshold
ndwi_mask <- ifel(NDWI>ndwi_threshold, 0, 1)

# Plot the NDWI mask
plot(ndwi_mask, main = "NDWI Mask")

# Set value 0 to NA to exclude the unwanted pixels
ndwi_mask <- ifel(ndwi_mask==0, NA, 1)
```

```r
# If desired, save the NDWI mask to a file
ndwi_threshold_modified <- gsub("\\.", "", ndwi_threshold)

ndwi_filename <- paste0(base_path,"/mask/",file_name_rectified,"_ndwi_mask_",ndwi_threshold_modified)
writeRaster(ndwi_mask, filename = file.path(ndwi_filename),
            filetype = "ENVI",
            gdal = "INTERLEAVE=BSQ",
            overwrite = TRUE,
            datatype = "INT1U")

################################################################################
# Create NDVI Mask NDVI = (NIR - Red) / (NIR + Red)
################################################################################

# Calculate the NDVI
NDVI <- (NIR_average-red_average)/(NIR_average+red_average)

# Plot histogram of the value distribution
hist(NDVI, breaks = seq(terra::minmax(NDVI)[1], terra::minmax(NDVI)[2] + 0.05, by = 0.01),
     main = "Histogram of NDVI", xlab = "NDVI")

# Create the NDVI mask (binary values) with a threshold of 0.1
#ndvi_threshold <- 0.3
ndvi_mask <- ifel(NDVI>ndvi_threshold, 1, 0)

# Plot the NDVI mask
plot(ndvi_mask, main = "NDVI Mask")

# Set value 0 to NA to exclude the unwanted pixels
ndvi_mask <- ifel(ndvi_mask==0, NA, 1)

# If desired, save the NDVI mask to a file
ndvi_threshold_modified <- gsub("\\.", "", ndvi_threshold)

ndvi_filename <- paste0(base_path,"/mask/",file_name_rectified,"_ndvi_mask_",ndvi_threshold_modified)
writeRaster(ndvi_mask, filename = file.path(ndvi_filename),
            filetype = "ENVI",
            gdal = "INTERLEAVE=BSQ",
            overwrite = TRUE,
            datatype = "INT1U")

################################################################################
# Create SAVI Mask SAVI = ((NIR - Red) / (NIR + Red + L)) * (1 + L)
################################################################################

# Set the L parameter for SAVI
L <- savi_L

# Calculate the SAVI
SAVI <- ((NIR_average - red_average) * (1 + L)) / (NIR_average + red_average + L)

# Plot a histogram of the SAVI values to inspect the distribution
hist(SAVI, breaks = seq(terra::minmax(SAVI)[1], terra::minmax(SAVI)[2] + 0.05, by = 0.01),
```

```
     main = "Histogram of SAVI", xlab = "SAVI")

# Create a SAVI mask (e.g., thresholding SAVI to identify vegetation)
# This threshold can be adjusted based on your analysis needs
#savi_threshold <- 0.2

savi_mask <- ifel(SAVI > savi_threshold, 1, 0)  # Here, 0.2 is an example threshold

# Plot the SAVI mask
plot(savi_mask, main = "SAVI Mask")

# Set value 0 to NA to exclude the unwanted pixels
savi_mask <- ifel(savi_mask==0, NA, 1)

# If desired, save the SAVI mask to a file
savi_threshold_modified <- gsub("\\.", "", savi_threshold)

savi_filename <- paste0(base_path,"/mask/",file_name_rectified,"_savi_mask_",savi_threshold_modified)
writeRaster(savi_mask, filename = file.path(savi_filename),
            filetype = "ENVI",
            gdal = "INTERLEAVE=BSQ",
            overwrite = TRUE,
            datatype = "INT1U")

################################################################################
# Create stacked Mask
################################################################################

mask <- mosaic(savi_mask, ndwi_mask, ndvi_mask, fun="min")

plot(mask)
mask <- ifel(mask==0, NA, 1)

# Now write the raster file
stack_filename <- paste0(base_path,"/mask/",file_name_rectified,"_stacked_mask")
writeRaster(mask, filename = file.path(stack_filename),
            filetype = "ENVI",
            gdal = "INTERLEAVE=BSQ",
            overwrite = TRUE,
            datatype = "INT1U")
```

Plot location species analysis

To estimate how many spectral species are to be expected, the plot locations lying withing the flight strips are examined. The count of the number of species that make up more than 1% ot the coverage is set as an input for the BiodivmapR calculations. 04_Species_analysis.R.

```
# clean environment
rm(list=ls(all=TRUE));gc()
graphics.off()

library(ggplot2)

# Define parameter script
```

```r
source('00_Project_Parameter.R')

csv_file_path <- paste0(base_path,'/data/species_analysis/',csv_file_name)

data <- read.csv(csv_file_path, header = TRUE, fileEncoding = "latin1")

species_data <- data[, -c(1, ncol(data))]  # Remove the first and last columns

# Calculate total abundance for each species
species_sums <- sort(colSums(species_data, na.rm = TRUE), decreasing = TRUE)

# Calculate relative abundance
total_abundance <- sum(species_sums)  # Total abundance across all species
relative_abundance <- species_sums / total_abundance * 100  # Convert to percentages

# Variance of each species across samples
species_variance <- apply(species_data, 2, var, na.rm = TRUE)

# Set threshold for significance (e.g., minimum relative abundance of 1%)
threshold <- 1
significant_species <- names(relative_abundance[relative_abundance >= threshold])
insignificant_species <- names(relative_abundance[relative_abundance < threshold])

# Calculate relative abundance
relative_abundance_df <- data.frame(
  Species = names(relative_abundance),
  RelativeAbundance = relative_abundance
)

# Calculate the number of species above the threshold
num_significant_species <- sum(relative_abundance >= threshold)

# Function to round up to the nearest multiple of n
round_up <- function(x, multiple) {
  ceiling(x / multiple) * multiple
}

nbclusters_calculated <- round_up(num_significant_species, 5)

# Create the plot with the additional annotation
relative_abundance_plot <- ggplot(relative_abundance_df, aes(x = reorder(Species, -RelativeAbundance), y
  geom_bar(stat = "identity", aes(fill = RelativeAbundance >= threshold), show.legend = FALSE) +
  scale_fill_manual(values = c("TRUE" = "steelblue", "FALSE" = "lightgray")) +
  geom_hline(yintercept = threshold, linetype = "dashed", color = "red", size = 1) +
  annotate("text", x = nrow(relative_abundance_df) / 2, y = max(relative_abundance) * 0.9,
           label = paste(num_significant_species, "species above", threshold, "% threshold"),
           color = "blue", size = 5, angle = 0, hjust = 0.5) +  # Annotation for number of significant
  annotate("text", x = nrow(relative_abundance_df) / 2, y = threshold + 0.5,
           label = paste("Threshold =", threshold, "%"), color = "red", size = 4, angle = 0, hjust = 0.5
  labs(
    title = "Relative Abundance of Species",
    x = "Species",
    y = "Relative Abundance (%)"
```

```r
  ) +
  theme_minimal() +
  theme(
    axis.text.x = element_text(angle = 45, hjust = 1),
    axis.line.x = element_line(color = "black")
  )

# Create the directory
if (!dir.exists("data/species_analysis/plots")) dir.create("species_analysis/plots", recursive = TRUE)


print(relative_abundance_plot)

# Save the relative abundance plot
ggsave(paste0(base_path,"/data/species_analysis/plots/relative_abundance_plot.png"), relative_abundance_



# Print summary
cat("Total species:", length(species_sums), "\n")
cat("Significant species (>= 1%):", length(significant_species), "\n")
cat("Insignificant species (< 1%):", length(insignificant_species), "\n\n")

cat("Significant Species:\n")
print(significant_species)



# Define threshold for cumulative percentage
cumulative_threshold <- 95  # You can change this value to any percentage

# Prepare the data for Pareto chart
species_df <- data.frame(
  Species = names(species_sums),
  Abundance = species_sums
)

# Sort species by abundance in descending order and calculate cumulative percentage
species_df <- species_df[order(-species_df$Abundance), ]
species_df$Cumulative <- cumsum(species_df$Abundance) / sum(species_df$Abundance) * 100

# Find the number of species required for the given cumulative threshold
n_species_threshold <- which(species_df$Cumulative >= cumulative_threshold)[1]  # First species to exce

# Create Pareto chart
pareto_plot <- ggplot(species_df, aes(x = reorder(Species, -Abundance), y = Abundance)) +
  # Bar chart
  geom_bar(stat = "identity", fill = "steelblue") +

  # Cumulative line
  geom_line(aes(y = (Cumulative / 100) * max(Abundance), group = 1), color = "red", size = 1) +
  geom_point(aes(y = (Cumulative / 100) * max(Abundance)), color = "red", size = 2) +
```

```
  # Threshold line
  geom_hline(yintercept = (cumulative_threshold / 100) * max(species_df$Abundance),
             linetype = "dashed", color = "darkgreen", size = 1) +

  # Annotate the number of species required for the threshold
  annotate("text", x = n_species_threshold,
           y = (cumulative_threshold / 100) * max(species_df$Abundance) * 0.9,
           label = paste(n_species_threshold, "species for", cumulative_threshold, "%"),
           color = "darkgreen", angle = 45, hjust = 1) +

  # Customize y-axis with dual axes
  scale_y_continuous(
    name = "Abundance",
    sec.axis = sec_axis(~ . / max(species_df$Abundance) * 100, name = "Cumulative Percentage")
  ) +

  # Titles and labels
  labs(title = paste("Pareto Chart of Species Abundance (", cumulative_threshold, "% Threshold)", sep =
       x = "Species", y = "Abundance") +

  # Rotate x-axis labels and add axis line
  theme_minimal() +
  theme(
    axis.text.x = element_text(angle = 45, hjust = 1),
    axis.line.x = element_line(color = "black")  # Add x-axis line
  )

print(pareto_plot)

# Save the Pareto chart
ggsave(paste0(base_path,"/data/species_analysis/plots/pareto_chart.png"), pareto_plot, dpi = 300, width
```

Spectral species analysis

The last script is run in four steps: 1. Perform dimensionality reduction using PCA 2. Select relevant PCs (for now a visual process using QGIS) 3. Perform the mapping of spectral species, α and β- diversity 4. Import the results to QGIS for visualisation

05_Data_Analysis.R

```
# clean environment
rm(list=ls(all=TRUE));gc()
graphics.off()

# load biodivMapR and useful libraries
library(biodivMapR)
library(labdsv)
library(tools)
library(ggplot2)
library(gridExtra)
library(terra)

# Define parameter script
source('00_Project_Parameter.R')
```

```
# input image folder path rectified
Datadir <- paste0(base_path,'/data/rectified')
# name of the image file
NameRaster <- file_name_rectified
Input_Image_File <- file.path(Datadir,NameRaster)
Input_HDR_File <- get_HDR_name(Input_Image_File,showWarnings = FALSE)




dir.create(path = Datadir,recursive = T,showWarnings = F)

################################################################################
##                       Set parameters for biodivMapR                        ##
## https://jbferet.github.io/biodivMapR/articles/biodivMapR_2.html            ##
################################################################################
# Define path for image file to be processed

# Define path for corresponding mask file
# Set to FALSE if no mask available
#Input_Mask_File <- FALSE
Input_Mask_File <- paste0(base_path,'/mask/',mask_name)
# Define path for master output directory where files produced during the process are saved

# Master output directory (remove unnecessary line break)
Output_Dir <- paste0(base_path,'/result')
dir.create(path = Output_Dir, recursive = TRUE, showWarnings = FALSE)

dir.create(path = Output_Dir,recursive = T,showWarnings = F)
# Define levels for radiometric filtering
NDVI_Thresh <- 0.3
Blue_Thresh <- 500
NIR_Thresh <- 1500
# Apply normalization with continuum removal?
Continuum_Removal <- FALSE
# Type of dimensionality reduction
TypePCA <- 'SPCA'
# PCA FILTERING:        Set to TRUE if you want second filtering based on PCA outliers to be processed.
# Slower process
# Automatically set to FALSE if TypePCA     = 'MNF'
FilterPCA <- FALSE
# window size for computation of spectral diversity
window_size <- 20
# computational parameters
nbCPU <- 4
MaxRAM <- 8
# number of clusters (spectral species)
nbclusters <- nbclusters_calculated

Excluded_WL <- c(0, 442)
Excluded_WL <- rbind(Excluded_WL, c(1368, 1499))
Excluded_WL <- rbind(Excluded_WL, c(1779, 2055))
```

```r
Excluded_WL <- rbind(Excluded_WL, c(2400, 2501))


################################################################################
##                  Perform PCA & Dimensionality reduction                    ##
## https://jbferet.github.io/biodivMapR/articles/biodivMapR_4.html            ##
################################################################################
print("PERFORM DIMENSIONALITY REDUCTION")
#debug(perform_PCA)
PCA_Output <- perform_PCA(Input_Image_File = Input_Image_File,
                          Input_Mask_File = Input_Mask_File,
                          Output_Dir = Output_Dir,
                          TypePCA = TypePCA,
                          FilterPCA = FilterPCA,
                          Excluded_WL = Excluded_WL,
                          nbCPU = nbCPU,
                          MaxRAM = MaxRAM,
                          Continuum_Removal = Continuum_Removal)


# Save the list as an RDS file
pca_output_rds_file_path = paste0(Output_Dir,"/",NameRaster,"/",TypePCA,"/PCA/","PCA_Output.rds")
saveRDS(PCA_Output, file = pca_output_rds_file_path)

# Later, load the list back into R
PCA_Output <- readRDS(pca_output_rds_file_path)


# path for the updated mask
Input_Mask_File <- PCA_Output$MaskPath

var_exp <- (PCA_Output$PCA_model$sdev^2/sum(PCA_Output$PCA_model$sdev^2))*100
barplot(var_exp, names.arg = colnames(PCA_Output$PCA_model$x))

pca_output_image_file_path = paste0(Output_Dir,"/",NameRaster,"/",TypePCA,"/PCA/","OutputPCA_30_PCs")
print(pca_output_image_file_path)
pca_output_image <- rast(pca_output_image_file_path)
plot(pca_output_image, main = "Principal Components", nc = 5, maxnl = 30)  # maxnl allows all 30 layers


# Define the path for PCA plots
# Define the path for individual PCA component plots
pca_plots_file_path <- paste0(Output_Dir, "/", NameRaster, "/", TypePCA, "/PCA/PCA_Plots")

# Create the directory if it doesn't exist
if (!dir.exists(pca_plots_file_path)) {
  dir.create(pca_plots_file_path, recursive = TRUE, showWarnings = FALSE)
}

# Initialize a list to store output plot file paths
output_plots <- list()

# Loop through each layer in the PCA output image
```

```r
for (i in 1:nlyr(pca_output_image)) {
  # Extract each PC layer as a separate raster
  pc_layer <- pca_output_image[[i]]

  # Define the output file path for the plot
  output_file <- paste0(pca_plots_file_path, "/PCA_PC_", i, ".png")

  # Save the plot as a PNG
  png(filename = output_file, width = 800, height = 600)
  plot(pc_layer,
       main = paste0("Principal Component ", i),
       col = terrain.colors(100))  # Customize colors if needed
  dev.off()  # Close the PNG device

  # Store the output file path
  output_plots[[i]] <- output_file
}

# Optional: print confirmation
print("PCA plots saved to:")
print(pca_plots_file_path)

# Select components from the PCA/SPCA/MNF raster
# Sel_PC = path of the file where selected components are stored
Sel_PC <- select_PCA_components(Input_Image_File = Input_Image_File,
                                Output_Dir = Output_Dir,
                                PCA_Files = PCA_Output$PCA_Files,
                                TypePCA = PCA_Output$TypePCA,
                                File_Open = TRUE)

################################################################################
##                  Perform Spectral species mapping                          ##
## https://jbferet.github.io/biodivMapR/articles/biodivMapR_5.html            ##
################################################################################
print("MAP SPECTRAL SPECIES")
Kmeans_info <- map_spectral_species(Input_Image_File = Input_Image_File,
                                    Input_Mask_File = PCA_Output$MaskPath,
                                    Output_Dir = Output_Dir,
                                    SpectralSpace_Output = PCA_Output,
                                    nbclusters = nbclusters,
                                    nbCPU = nbCPU, MaxRAM = MaxRAM)

################################################################################
##                Perform alpha and beta diversity mapping                    ##
## https://jbferet.github.io/biodivMapR/articles/biodivMapR_6.html            ##
################################################################################
print("MAP ALPHA DIVERSITY")
Index_Alpha   = c('Shannon','Simpson')
#Index_Alpha <- c('Shannon')
map_alpha_div(Input_Image_File = Input_Image_File,
              Output_Dir = Output_Dir,
              TypePCA = TypePCA,
              window_size = window_size,
```

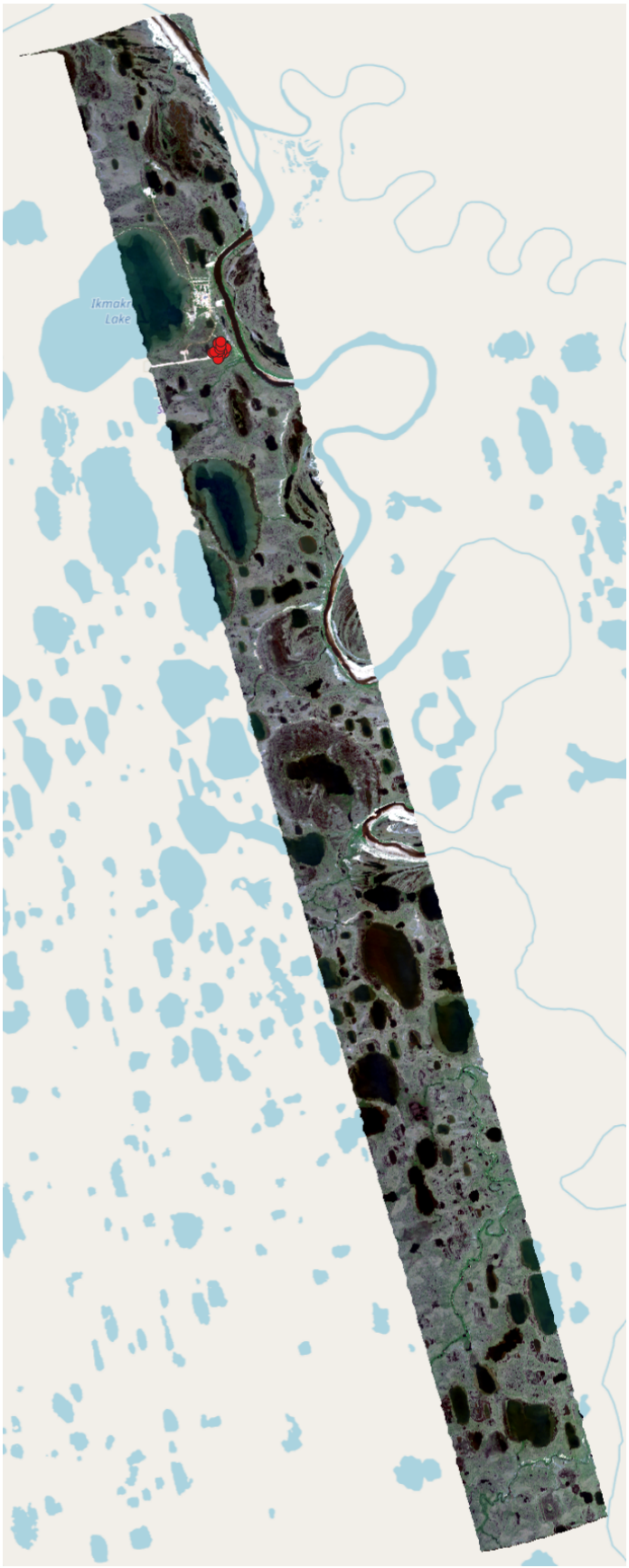
```
               nbCPU = nbCPU,
               MaxRAM = MaxRAM,
               Index_Alpha = Index_Alpha,
               nbclusters = nbclusters)

print("MAP BETA DIVERSITY")
map_beta_div(Input_Image_File = Input_Image_File,
             Output_Dir = Output_Dir,
             TypePCA = TypePCA,
             window_size = window_size,
             nbCPU = nbCPU,
             MaxRAM = MaxRAM,
             nbclusters = nbclusters)
```

Sample of a flight strip in subzone d (Atqasuk Airport)

RGB and SAVI

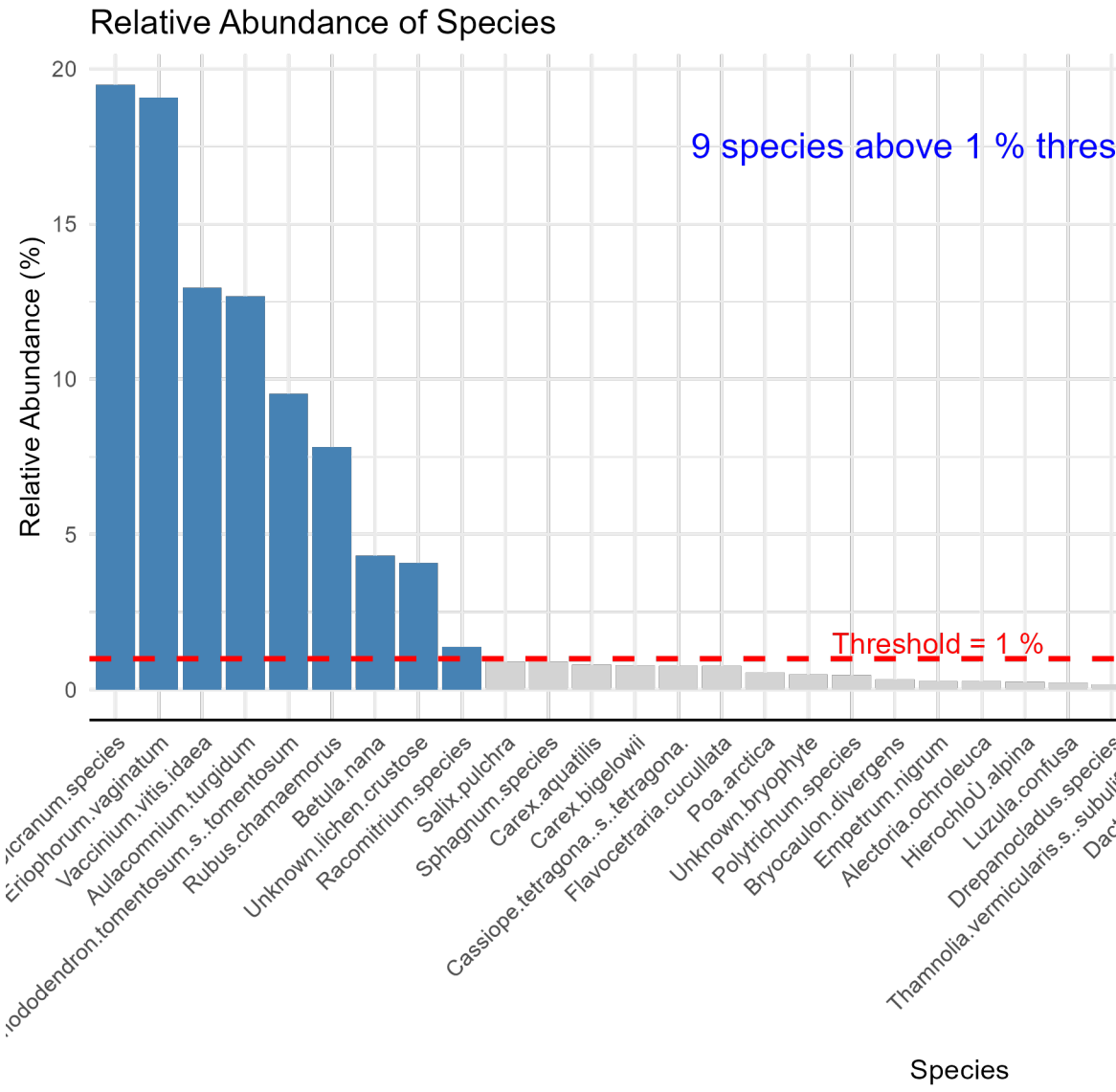
RGB	SAVI

RGB	SAVI
	

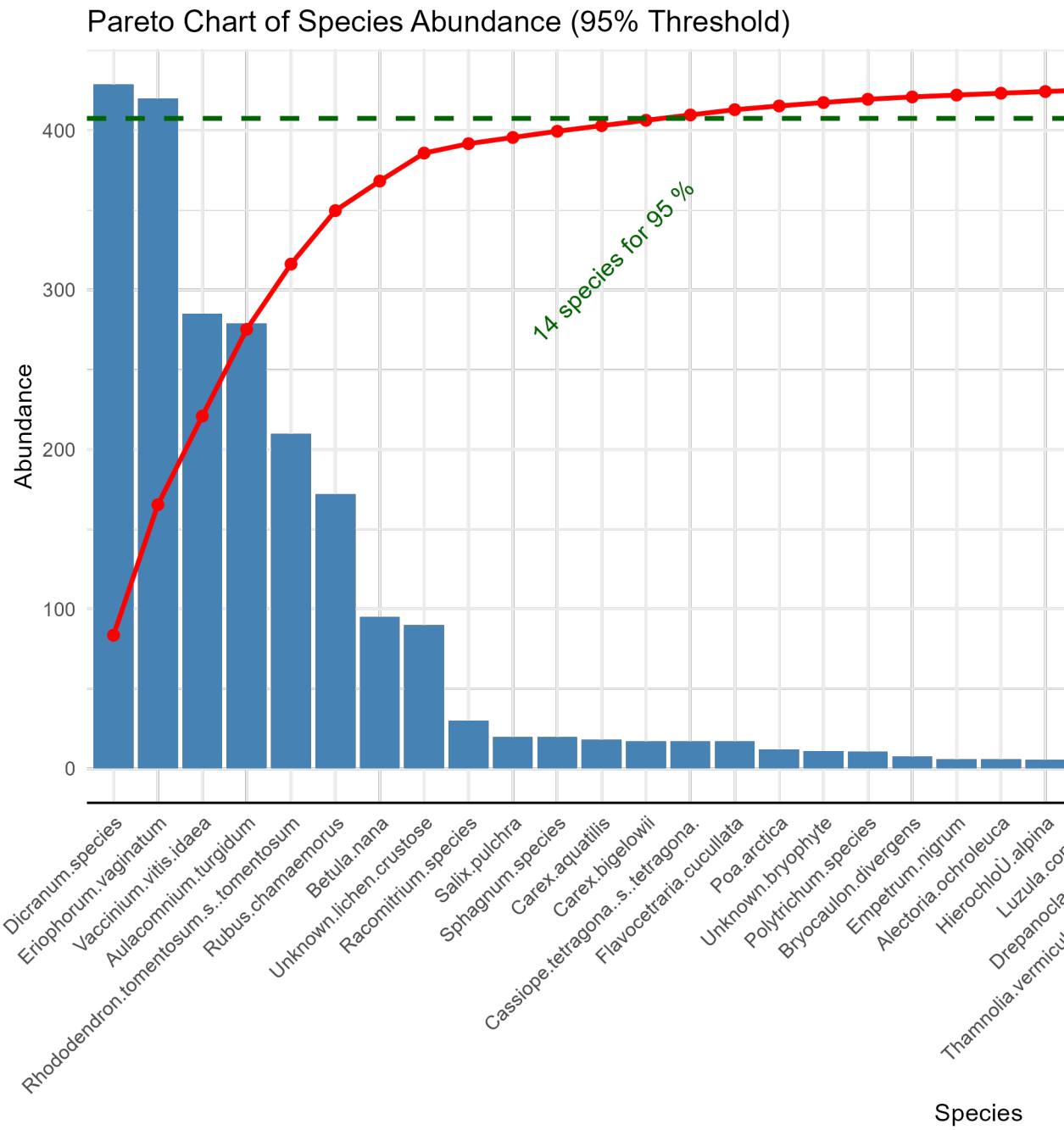
RGB	SAVI
	

RGB	SAVI

Spectral species count



Relative abundance



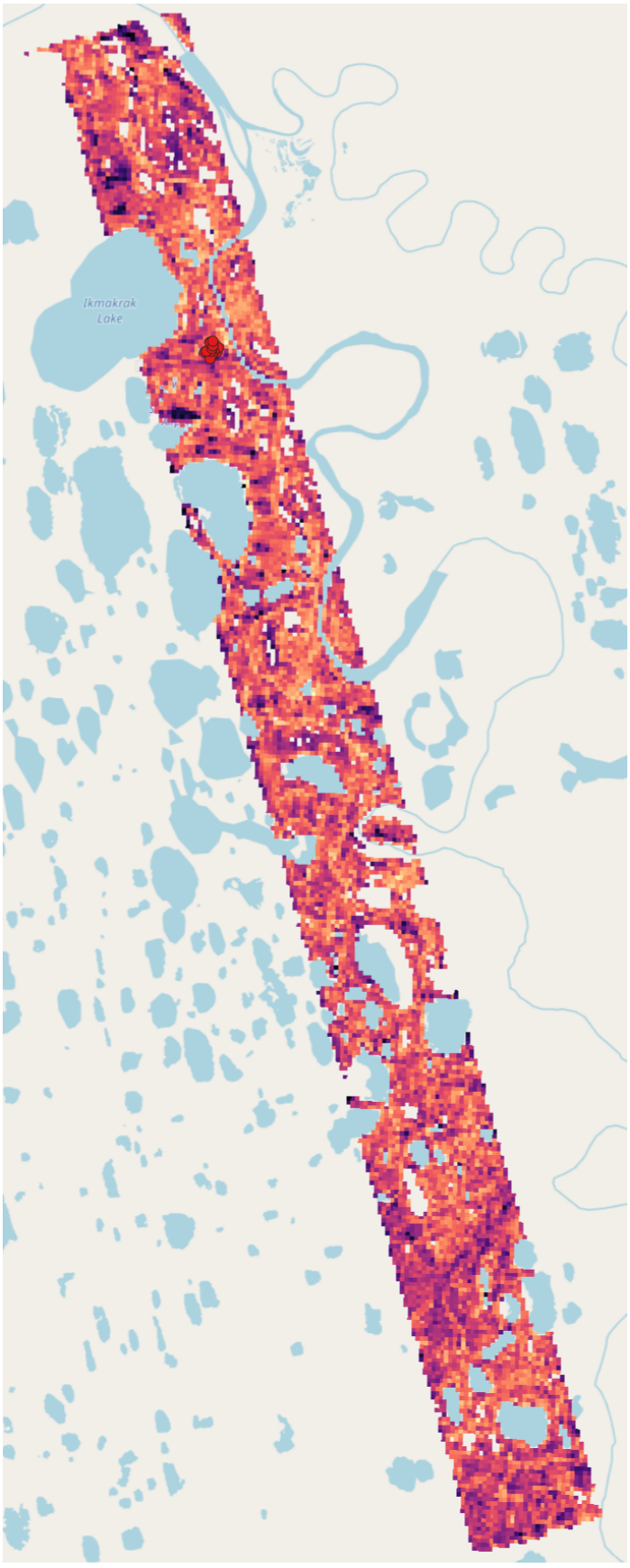
Pareto chart

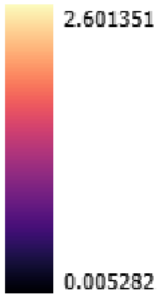
Diversity calculations

Alpha

Shannon diversity	Index

Shannon diversity	Index

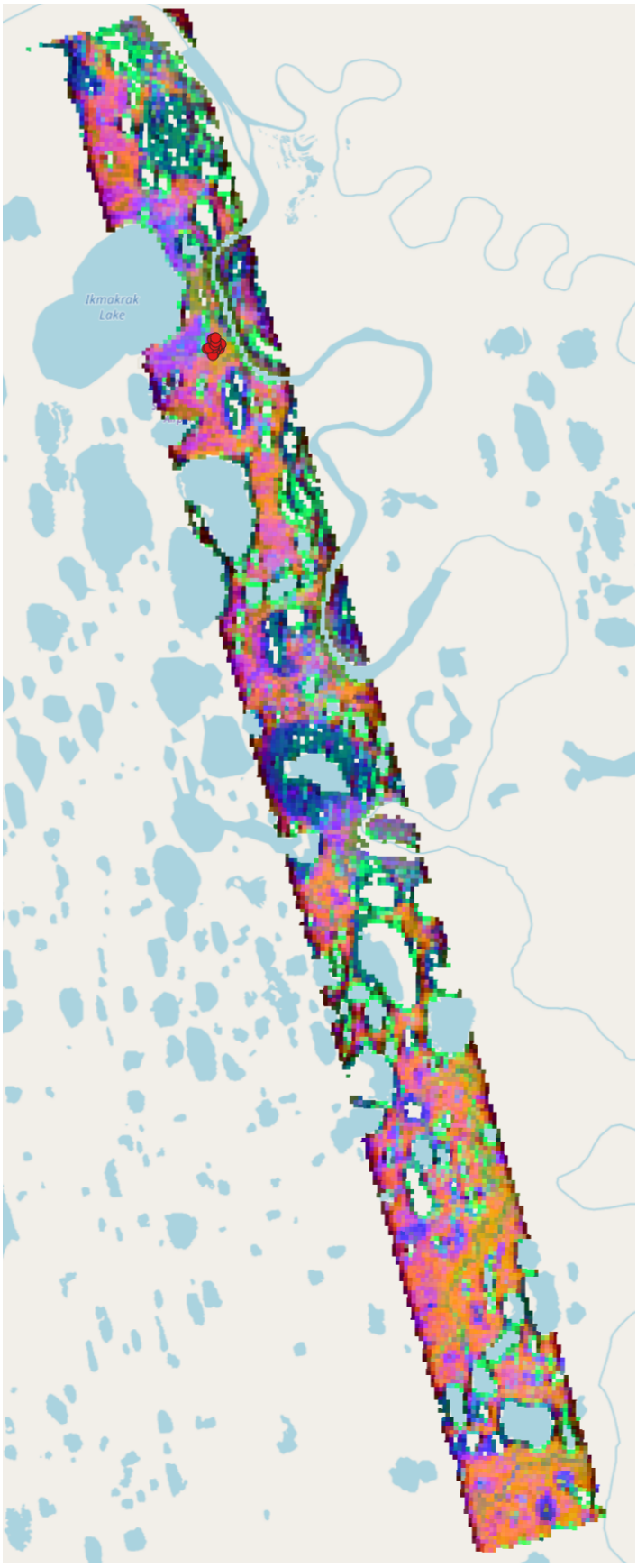


Shannon diversity	Index
	

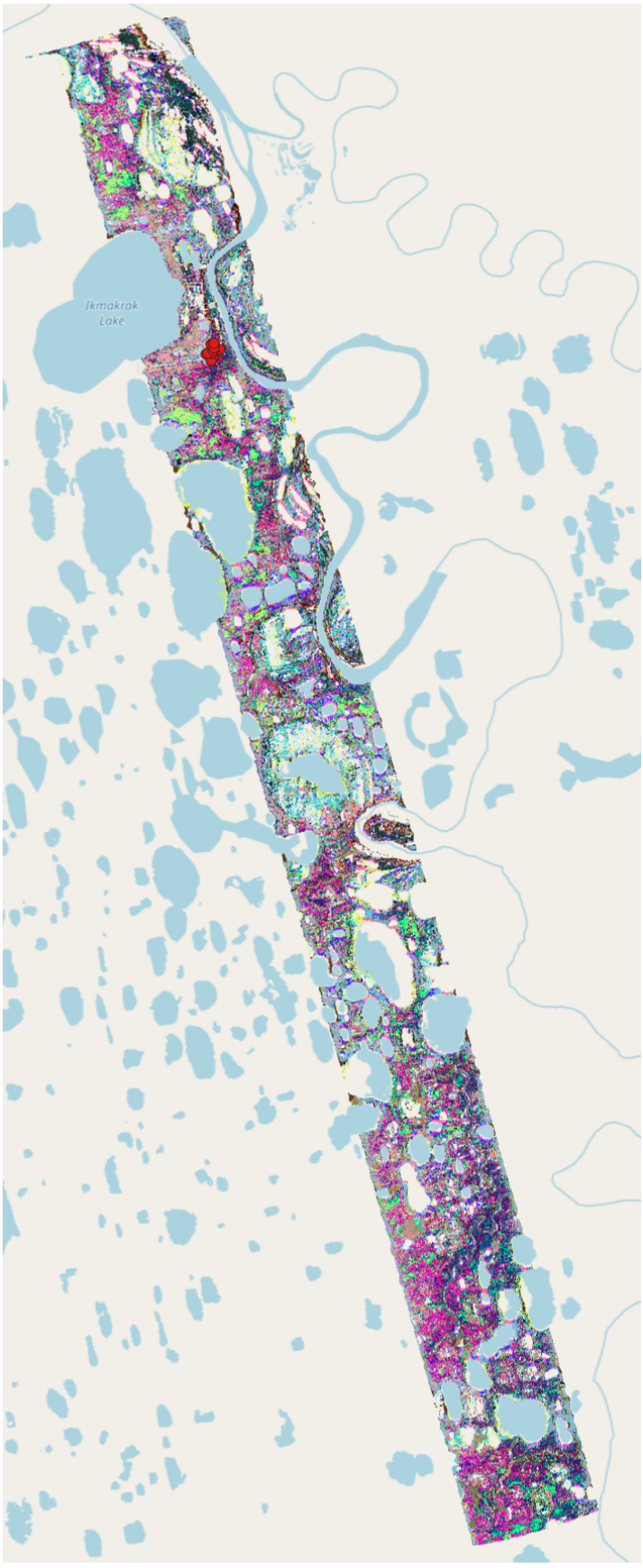
Beta

Beta	Spectral Species

Beta	Spectral Species



Beta	Spectral Species



Beta	Spectral Species