EDS 223: week 8 lab

Ruth Oliver

2023-11-20

Overview

Phenology is the timing of life history events. Important phenological events for plants involve the growth of leaves, flowering, and senescence (death of leaves). Plants species adapt the timing of these events to local climate conditions to ensure successful reproduction. Subsequently, animal species often adapt their phenology to take advantage of food availability. As the climate shifts this synchronization is being thrown out of whack. Shifts in phenology are therefore a common yardstick of understanding how and if ecosystems are adjusting to climate change.

Plant species may employ the following phenological strategies:

- winter deciduous: lose leaves in the winter, grow new leaves in the spring
- drought deciduous: lose leaves in the summer when water is limited
- evergreen: maintain leaves yearround

credit: this lab is based on a materials developed by Chris Kibler.

Task

In this lab we are analyzing plant phenology near the Santa Clara River which flows from Santa Clarita to Ventura. We will investigate the phenology of the following plant communities:

- riparian forests: grow along the river, dominated by winter deciduous cottonwood and willow trees
- grasslands: grow in openspaces, dominated by drought deciduous grasses
- chaparral shrublands: grow in more arid habitats, dominated by evergreen shrubs

To investigate the phenology of these plant communities we will a time series of Landsat imagery and polygons identifying the locations of study sites within each plant community.

Data

Landsat Operational Land Imager (OLI sensor)

- 8 pre-processed scenes
  - Level 2 surface reflectance products
  - erroneous values were set to NA
  - scale factor set to 100
  - bands 2-7
  - dates in filenname

Study sites

- polygons representing sites
  - study_site: character string with plant type

Summary of approach

- Convert spectral relfectance into a measure of vegetation productivity (NDVI)
- Calulate NDVI throughout the year
- Summarize NDVI values within vegetation communities
- Visualize changes in NDVI within vegetation communities

Workflow

```
library(terra)
library(sf)
library(dplyr)
library(tidyr)
library(stringr)
library(ggplot2)
library(here)
library(tmap)
library(cowplot)

rm(list = ls())
here::i_am("labs/week8.Rmd")
```

create NDVI function

Let's start by defining a function to compute the Normalized Difference Vegetation Index (NDVI). NDVI computes the difference in reflectance in the near infrared and red bands, normalized by their sum.

```
ndvi_fun = function(nir, red){
  (nir - red) / (nir + red)
}
```

computing NDVI for a single scene

We have 8 scenes collected by the Landsat OLI sensor on 8 different days throughout the year. Let's start by loading in the first scene collected on June 12, 2018

```
landsat_20180612 <-rast(here("labs", "data", "week8", "landsat_20180612.tif"))

landsat_20180612
```

Now let's update the names of the layers to match the spectral bands they correspond to

```
names(landsat_20180612) <- c("blue", "green", "red", "NIR", "SWIR1", "SWIR2")

landsat_20180612
```

Now we can apply the NDVI function we created to compute NDVI for this scene using the `lapp()` function. The `lapp()` function applies a function to each cell using layers as arguments. Therefore, we need to tell `lapp()` which layers (or bands) to pass into the function. The NIR band is the 4th layer and the red band is the 3rd layer in our raster. In this case, because we defined the NIR band as the first argument and the red band as the second argument in our function, we tell `lapp()` to use the 4th layer first and 3rd layer second.

```
ndvi_20180612 <- lapp(landsat_20180612[[c(4, 3)]], fun = ndvi_fun)

ndvi_20180612
```

computing NDVI for all scences - attempt 1

Now we want to repeat the same operations for all 8 scenes. Below is a possible solution, but it's pretty clunky.

```
landsat_20180612 <-rast(here("labs", "data", "week8", "landsat_20180612.tif"))
landsat_20180815 <- rast(here("labs", "data", "week8", "landsat_20180815.tif"))
landsat_20181018 <- rast(here("labs", "data", "week8", "landsat_20181018.tif"))
landsat_20181103 <- rast(here("labs", "data", "week8", "landsat_20181103.tif"))
landsat_20190122 <- rast(here("labs", "data", "week8", "landsat_20190122.tif"))
landsat_20190223 <- rast(here("labs", "data", "week8", "landsat_20190223.tif"))
landsat_20190412 <- rast(here("labs", "data", "week8", "landsat_20190412.tif"))
landsat_20190701 <- rast(here("labs", "data", "week8", "landsat_20190701.tif"))
```

load in each layer

```
names(landsat_20180612) <- c("blue", "green", "red", "NIR", "SWIR1", "SWIR2")
names(landsat_20180815) <- c("blue", "green", "red", "NIR", "SWIR1", "SWIR2")
names(landsat_20181018) <- c("blue", "green", "red", "NIR", "SWIR1", "SWIR2")
names(landsat_20181103) <- c("blue", "green", "red", "NIR", "SWIR1", "SWIR2")
names(landsat_20190122) <- c("blue", "green", "red", "NIR", "SWIR1", "SWIR2")
names(landsat_20190223) <- c("blue", "green", "red", "NIR", "SWIR1", "SWIR2")
names(landsat_20190412) <- c("blue", "green", "red", "NIR", "SWIR1", "SWIR2")
names(landsat_20190701) <- c("blue", "green", "red", "NIR", "SWIR1", "SWIR2")
```

rename each layer

```
ndvi_20180612 <- lapp(landsat_20180612[[c(4, 3)]], fun = ndvi_fun)
ndvi_20180815 <- lapp(landsat_20180815[[c(4, 3)]], fun = ndvi_fun)
ndvi_20181018 <- lapp(landsat_20181018[[c(4, 3)]], fun = ndvi_fun)
ndvi_20181103 <- lapp(landsat_20181103[[c(4, 3)]], fun = ndvi_fun)
ndvi_20190122 <- lapp(landsat_20190122[[c(4, 3)]], fun = ndvi_fun)
ndvi_20190223 <- lapp(landsat_20190223[[c(4, 3)]], fun = ndvi_fun)
ndvi_20190412 <- lapp(landsat_20190412[[c(4, 3)]], fun = ndvi_fun)
ndvi_20190701 <- lapp(landsat_20190701[[c(4, 3)]], fun = ndvi_fun)
```

compute NDVI for each layer

combine NDVI layers Now we can combine the NDVI layers we created into a single raster stack.

```
all_ndvi <- c(ndvi_20180612,
              ndvi_20180815,
              ndvi_20181018,
              ndvi_20181103,
              ndvi_20190122,
              ndvi_20190223,
              ndvi_20190412,
              ndvi_20190701)
```

Let's update the names of each layer to match the date of each image.

```
names(all_ndvi) <- c("2018-06-12",
                     "2018-08-15",
                     "2018-10-18",
                     "2018-11-03",
                     "2019-01-22",
                     "2019-02-23",
                     "2019-04-12",
                     "2019-07-01")
```

computing NDVI for all scenes - attempt 2

The first attempt was pretty clunky and required a lot of copy/pasting. Because we're performing the same operations over and over again, this is a good opportunity to generalize our workflow into a function!

starting from scratch Let's start over and see how we could do this more efficiently. We'll clear our environment and redefine our function for NDVI.

```
rm(list = ls())
here::i_am("labs/week8.Rmd")

ndvi_fun = function(nir, red){
  (nir - red) / (nir + red)
}
```

outlining our function Let's first sketch out what operations we want to perform so we can figure out what our function needs.

```
# note: this code is not meant to run! we're just outlining the function we want to create

create_ndvi_layer <- function(){
  landsat <- rast(file) # read in scene
  names(landsat) <- c("blue", "green", "red", "NIR", "SWIR1", "SWIR2") # rename layers
  ndvi <- lapp(landsat[[c(4, 3)]], fun = ndvi_fun) # compute NDVI
}

# what do we notice as what we need to pass into our function?
```

creating a list of files/scenes We want a list of the scenes so that we can tell our function to compute NDVI for each. To do that we look in our data folder for the relevant file.

```
# here we're asking for the names of all the files in the week8 folder
# the "pattern" option returns names that end in .tif (the file extension for the landsat scences)
# the "full.names" option returns the full file path for each scene

files <- list.files(here("labs", "data", "week8"), pattern = "*.tif", full.names = TRUE)

files
```

updating our function Now let's update our function to work with list of file names we created

```
# now we're passing our function a number that will correspond to the index in the list of file names
create_ndvi_layer <- function(i){
  landsat <- rast(files[i])
  names(landsat) <- c("blue", "green", "red", "NIR", "SWIR1", "SWIR2")
  ndvi <- lapp(landsat[[c(4, 3)]], fun = ndvi_fun)
}

# let's test our function by asking it to read in the first file
test <- create_ndvi_layer(1)
```

run function on all scenes! Now we can use our function to create a NDVI layer for each scene and stack them into a single rasterstack.

```
# create NDVI layer for each scene (day) and stack into a single rasterstack
all_ndvi <- c(create_ndvi_layer(1),
              create_ndvi_layer(2),
              create_ndvi_layer(3),
              create_ndvi_layer(4),
              create_ndvi_layer(5),
              create_ndvi_layer(6),
              create_ndvi_layer(7),
              create_ndvi_layer(8))

# update layer names to match date
names(all_ndvi) <- c("2018-06-12",
                     "2018-08-15",
                     "2018-10-18",
                     "2018-11-03",
                     "2019-01-22",
                     "2019-02-23",
                     "2019-04-12",
                     "2019-07-01")
```

Compare NDVI across vegetation communities

Now that we have computed NDVI for each of our scenes (days) we want to compare changes in NDVI values across different vegetation communities.

read in study sites First, we'll read in a shapefile of study sites.

```
sites <- st_read(here("labs","data","week8","study_sites.shp"))

sites

# plot study sites on a single NDVI layer
tm_shape(all_ndvi[[1]]) +
  tm_raster() +
  tm_shape(sites) +
  tm_polygons()
```

extract NDVI at study sites Here we find the average NDVI within each study site. The output of extract is a data frame with rows that match the study site dataset, so we bind the results to the original dataset.

```
sites_ndvi <- terra::extract(all_ndvi, sites, fun = "mean")

sites_annotated <- cbind(sites, sites_ndvi)
```

clean results We're done! Except our data is very untidy... Let's tidy it up!

- convert to data frame
- turn from wide to long format

- turn layer names into date format

```
sites_clean <- sites_annotated %>%
  st_drop_geometry() %>%
  select(-ID) %>%
  pivot_longer(!study_site) %>%
  rename("NDVI" = value) %>%
  mutate("year" = str_sub(name, 2, 5),
         "month" = str_sub(name, 7, 8),
         "day" = str_sub(name, -2, -1)) %>%
  unite("date", 4:6, sep = "-") %>%
  mutate("date" = lubridate::as_date(date))
```

```
ggplot(sites_clean,
       aes(x = date, y = NDVI,
           group = study_site, col = study_site)) +
  scale_color_manual(values = c("#EAAC8B", "#315C2B", "#315C2B", "#315C2B","#9EA93F")) +
  geom_line() +
  theme_minimal() +
  labs(x = "", y = "Normalized Difference Vegetation Index (NDVI)", col = "Vegetation type",
       title = "Seasonal cycles of vegetation productivity")
```

plot results