Assignment 14

Jeff Oliver

2025-04-29

Assignment Details

Purpose

The goal of this assignment is to practice working with geospatial raster and vector data.

Task

Write R code to successfully answer each question below.

Criteria for Success

- Code is within the provided code chunks or new code chunks are created where necessary
- Code chunks run without errors
- Code chunks have brief comments indicating which code is answering which part of the question
- Code will be assessed as follows:
  - Produces the correct answer using the requested approach: 100%
  - Generally uses the right approach, but a minor mistake results in an incorrect answer: 90%
  - Attempts to solve the problem and makes some progress using the core concept, but returns the wrong answer and does not demonstrate comfort with the core concept: 50%
  - Answer demonstrates a lack of understanding of the core concept: 0%
- Any questions requiring written answers are answered with sufficient detail

Due Date

May 6 at midnight MST

Assignment Exercises

1. Working with Raster Data in `terra` (20 pts)

In this exercise, you will work with the same precipitation data as in class, but this time focus in on the Himalayan region of Asia.

a. Load in the terra package. What version does it say you are using?

b. Load in the precipitation data ("global_precipitation.tif"), what is the EPSG used in these data?

c. Define a geographic extent object for the Himalayan region.

d. Crop the precipitation data to the Himalayan extent; what are resulting dimensions?

```
# 1a
# When loading terra library, will print version number to console
library(terra)
```

```
## terra 1.8.42
```

```
# 1b
# Read in data with rast
prec <- rast("image_files/global_precipitation.tif")
# Print prec raster to see EPSG (4326)
prec
```

```
## class       : SpatRaster
## dimensions  : 4320, 8640, 1  (nrow, ncol, nlyr)
## resolution  : 0.04166667, 0.04166667  (x, y)
## extent      : -180, 180, -90, 90  (xmin, xmax, ymin, ymax)
## coord. ref. : lon/lat WGS 84 (EPSG:4326)
## source      : global_precipitation.tif
## name        : global_precipitation
## min value   :                   0
## max value   :               11246
```

```
# 1c
# Geographic extent of Himalayas, roughly, is (lat, lon)
# Top left: 40.37, 69.33
# Bottom right: 26.93, 97.15
# ext takes arguments in this order: xmin, xmax, ymin, ymax
# or, in human: left, right, bottom, top
hima_ext <- ext(c(69.33, 97.15, 26.93, 40.37))

# 1d
# Crop precipitation to Himalayan extent
prec_hima <- crop(prec, hima_ext)
# Print new raster to see extent
prec_hima
```

```
## class       : SpatRaster
## dimensions  : 323, 668, 1  (nrow, ncol, nlyr)
## resolution  : 0.04166667, 0.04166667  (x, y)
## extent      : 69.33333, 97.16667, 26.91667, 40.375  (xmin, xmax, ymin, ymax)
## coord. ref. : lon/lat WGS 84 (EPSG:4326)
## source(s)   : memory
## varname     : global_precipitation
## name        : global_precipitation
## min value   :                  11
## max value   :                4853
```

2. Printing Maps (20 points)

Still working with the precipitation data, this exercise focuses on visualizing the data.

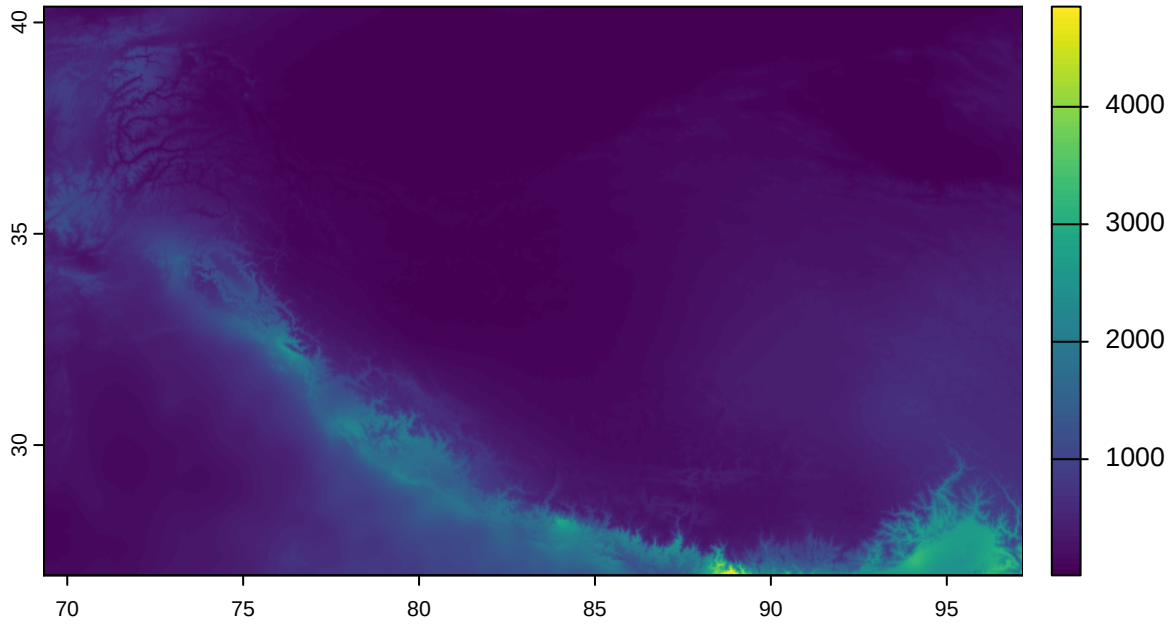
a. Plot the precipitation data for the Himalayan region.

b. Add a title to the plot.

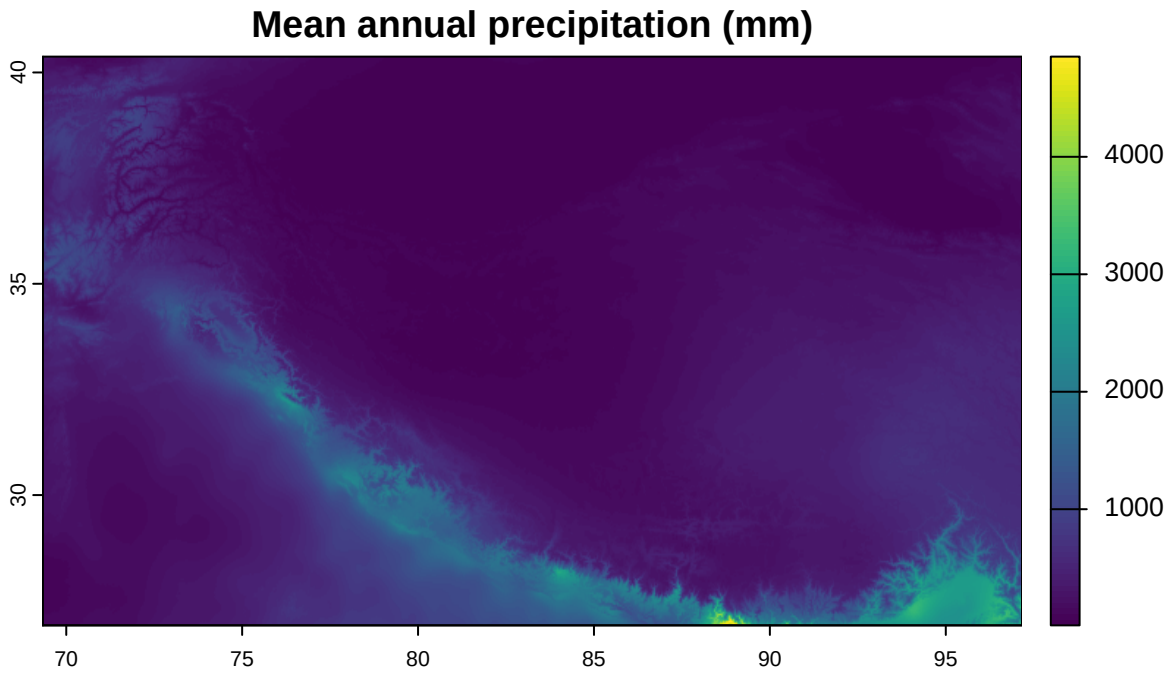
c. Change the palette to that of `topo.colors` (use `?topo.colors` for details) and print the resultant plot.

d. Change the palette to one from ColorBrewer for sequential (continuous) data.

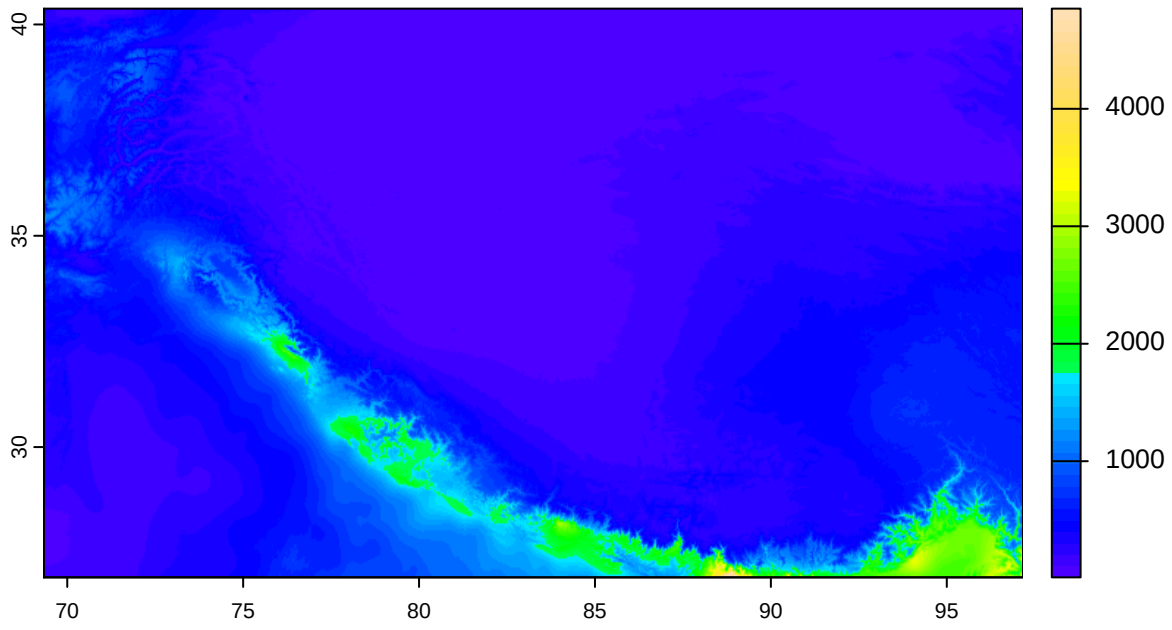
```
# 2a
# Plot the cropped precipitation data
plot(prec_hima)
```



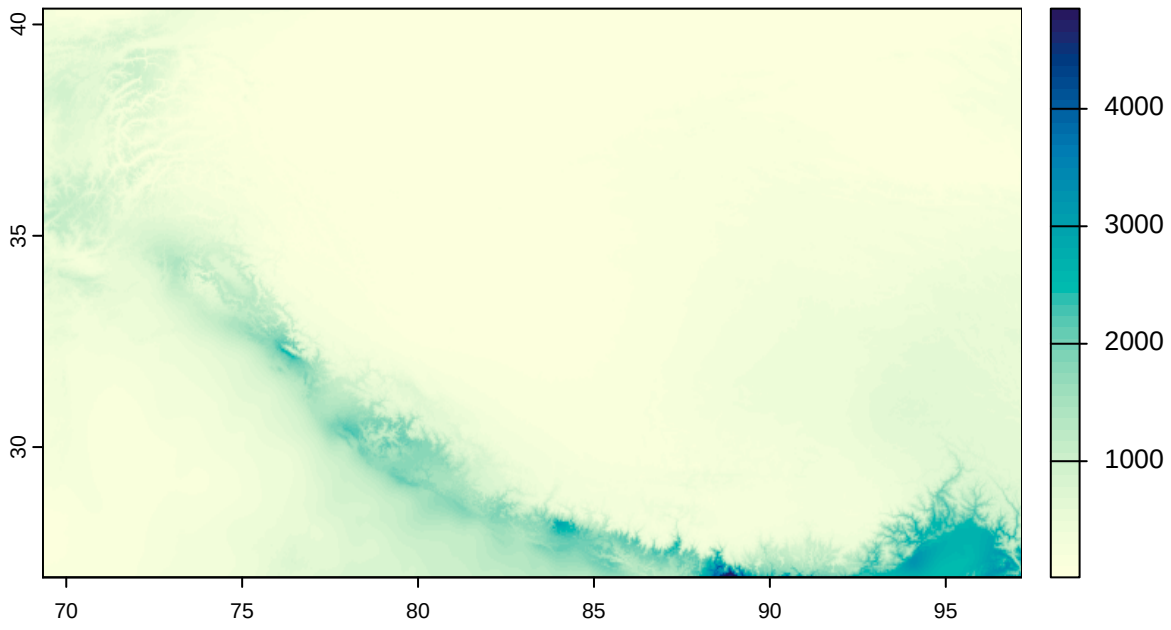

```
# 2b
# Plot the same data, this time adding a title
plot(prec_hima, main = "Mean annual precipitation (mm)")
```




```
# 2c
# Define the palette first, using the topo.colors function
prec_cols <- topo.colors(n = 50)
# Plot the map with those new colors
plot(prec_hima, col = prec_cols)
```




```
# 2d
# Define palette with hcl.colors, several options from Color Brewer, example
# below uses Yellow-Green-Blue palette, reversed so blue is highest
prec_cols <- rev(hcl.colors(n = 50, palette = "YlGnBu"))
plot(prec_hima, col = prec_cols)
```



3. Working with vector data in terra (20 points)

a. Create a data frame of three peaks in the Himalayan region: (lat, lon)

- K2: 35.93, 76.50
- Annapurna: 28.53, 83.81
- Mount Everest: 28.05, 86.91

b. Create a SpatVector object from the data frame with peak data.

A quick note about plotting with ***terra****:*

When you run multiple plot functions (or a similar function adding something onto the plot, such as `lines()` or `text()`), you need to run all of the functions together at the same time. Running the lines one by one will give you an error like "Error in graphics. . . plot.new has not been called yet."

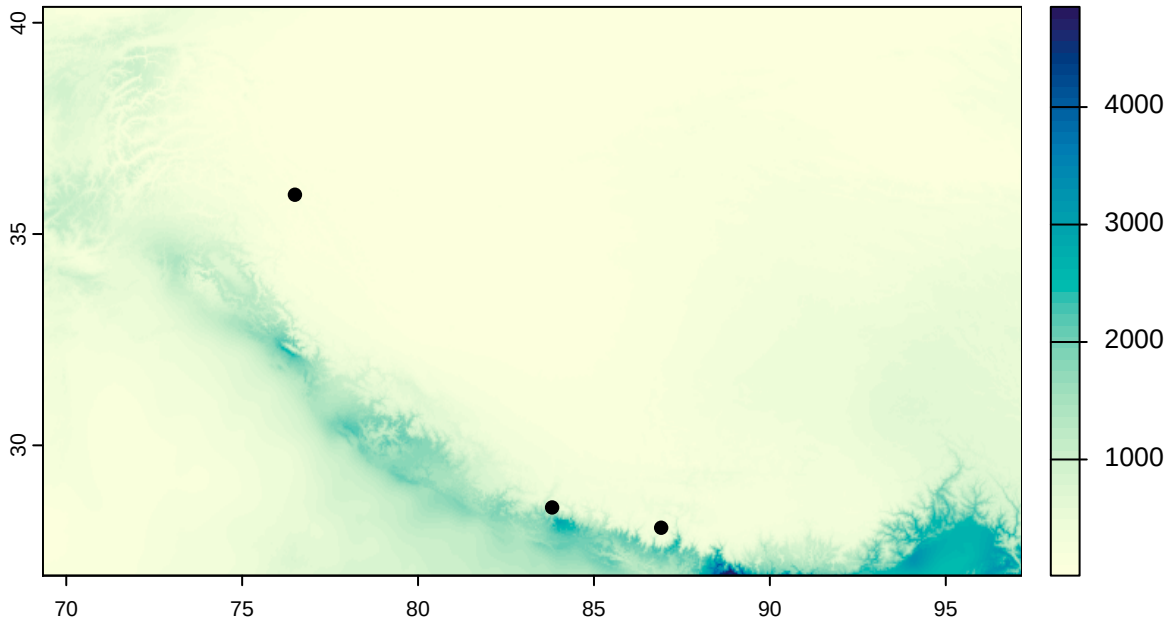
You can either highlight all likes of code creating the plot and run them together or use the green arrow at the top of the code chunk–either option should work.

c. Plot the precipitation data, and ddd the points of the peaks onto the plot.

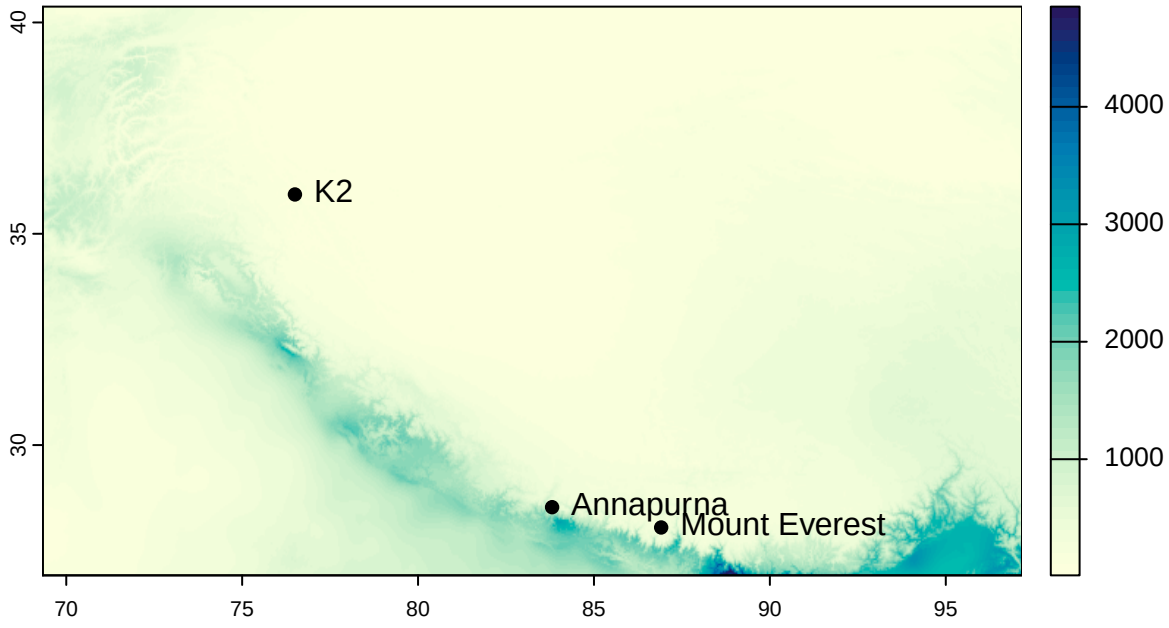
d. Add labels to the points representing the peaks.

```
# 3a
# Create data frame for three peaks
peaks <- data.frame(peak = c("K2", "Annapurna", "Mount Everest"),
                    lat = c(35.93, 28.53, 28.05),
                    lon = c(76.50, 83.81, 86.91))

# 3b
```

```r
# Convert data frame to SpatVector object
peaks_vect <- vect(peaks, crs = crs(prec_hima))

# 3c
# Draw base precipitation plot
plot(prec_hima, col = prec_cols)
plot(peaks_vect, add = TRUE)
```




```r
# 3d
# Add lines with lines function
plot(prec_hima, col = prec_cols)
plot(peaks_vect, add = TRUE)
text(peaks_vect, labels = "peak", pos = 4)
```



4. Modifying raster values (20 points)

a. Load in the elevation data ("global_elevation.tif"), what are the units for these data?

b. Convert the units of the raster to feet.

c. Crop the elevation data to the same geographic extent as precipitation data.

d. Plot the Himalayan elevation data.

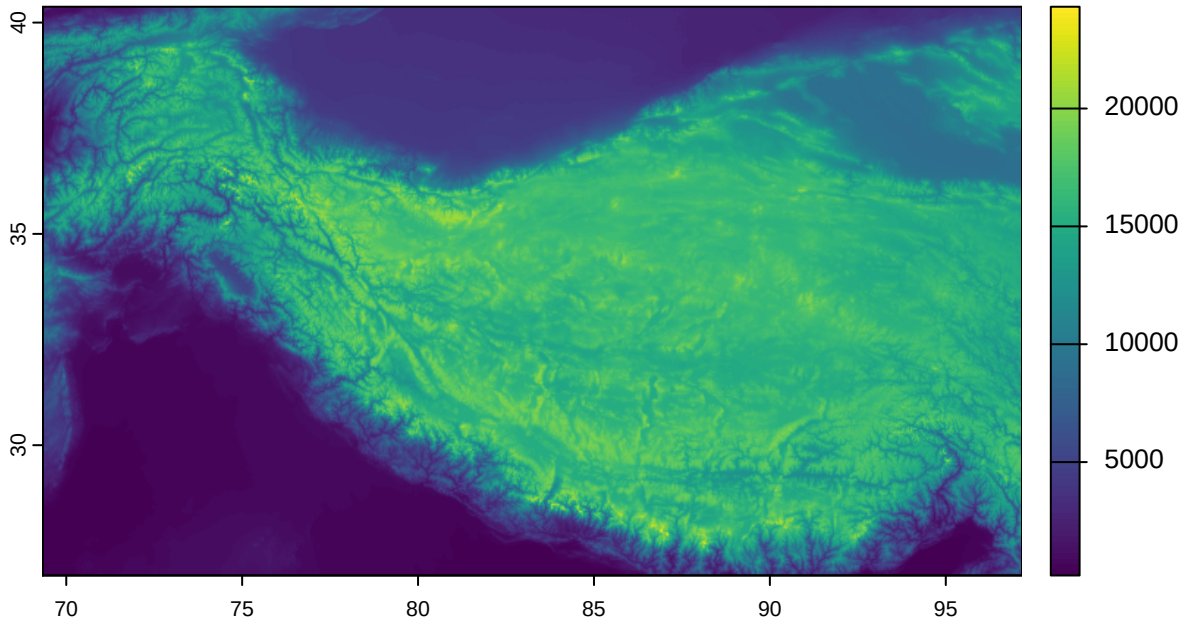
```
# 4a
# Load the data
elev <- rast("image_files/global_elevation.tif")
# Print the object to see range of elevations, safe to infer units are meters
elev
```

```
## class       : SpatRaster
## dimensions  : 4320, 8640, 1  (nrow, ncol, nlyr)
## resolution  : 0.04166667, 0.04166667  (x, y)
## extent      : -180, 180, -90, 90  (xmin, xmax, ymin, ymax)
## coord. ref. : lon/lat WGS 84 (EPSG:4326)
## source      : global_elevation.tif
## name        : global_elevation
## min value   :            -415
## max value   :            7412
```

```
# 4b
# 1 foot = 3.28084 meters
elev <- elev * 3.28084
```

```
## |---------|---------|---------|---------|=======================================
```

```
# 4c
# Crop to same geographic extent as for precipitation data
elev_hima <- crop(elev, hima_ext)

# 4d
plot(elev_hima)
```



5. Converting raster objects to spatial vector objects (20 points)

For this last exercise, create a plot that shows precipitation for the Himalayan region, using a color palette from ColorBrewer AND add a layer that is partially (50%) transparent to shade out regions below 13,000 feet. Take a look at the arguments for the `polys` function for how to set the color and transparency.

```
# 5a
# Define elevation threshold
alpine_min <- 13000

# 5b
# Create logical raster of high elevation (TRUE/FALSE)
elev_alpine <- elev_hima >= alpine_min

# 5c
# Convert all TRUE values to NA (want the polygon to shade out areas of low
# elevation, which are FALSE in this case)
elev_alpine[elev_alpine] <- NA
```

```r
# 5d
# Convert the raster with FALSE values only at places below 13,000 feet
low_elev_poly <- as.polygons(elev_alpine)

# 5e
# Plot the precipitation map
plot(prec_hima, col = prec_cols)
# Add black polygon with 50% transparency to shade out low-elevation sites
polys(x = low_elev_poly, col = "black", border = NA, alpha = 0.5)
```