Chapter 7

Norman Lim

2025-10-31

Hands-On: Working with Your First Raster

Preparing the environment:

Step 1: Load necessary packages

```
pacman::p_load(terra, sf, tidyverse, geodata)
```

Step 2: Set theme to make it easier to create consistent plots later.

```
theme_set(theme_minimal())
```

Loading Raster Data

Step 1: Load raster data from `geodata`:

```
global_elevation_10min <- geodata::elevation_global(
  res = 10, # 10 arc-minite resolution (coarse, smaller file)
  path = tempdir()
)
```

Step 2: Inspect the key properties of the raster data:

```
global_elevation_10min
```

```
## class       : SpatRaster
## size        : 1080, 2160, 1  (nrow, ncol, nlyr)
## resolution  : 0.1666667, 0.1666667  (x, y)
## extent      : -180, 180, -90, 90  (xmin, xmax, ymin, ymax)
## coord. ref. : lon/lat WGS 84 (EPSG:4326)
## source      : wc2.1_10m_elev.tif
## name        : wc2.1_10m_elev
## min value   :           -352
## max value   :           6251
```

Inspecting Your Raster (using terra functions)

Step 1: Check the resolution using `terra::res()`:

```r
terra::res(global_elevation_10min)
```

```
## [1] 0.1666667 0.1666667
```

Step 2: Check extent (bounding box) using `terra::ext()`:

```r
terra::ext(global_elevation_10min)
```

```
## SpatExtent : -180, 180, -90, 90 (xmin, xmax, ymin, ymax)
```

Step 3: Check coordinate system using `terra::crs()`:

```r
terra::crs(global_elevation_10min)
```

```
## [1] "GEOGCRS[\"WGS 84\",\n    ENSEMBLE[\"World Geodetic System 1984 ensemble\",\n        MEMBER[\"Wo
```

Step 4: Check number of layers/bands using `terra::nlyr()`:

```r
terra::nlyr(global_elevation_10min)
```

```
## [1] 1
```

Step 5: Look at some pixel values using `terra::values()` together with `head()`:

```r
head(terra::values(global_elevation_10min))[, 1]
```

```
## [1] NA NA NA NA NA NA
```

Step 6: Get min/max elevation values efficiently using `terra::minmax()`:

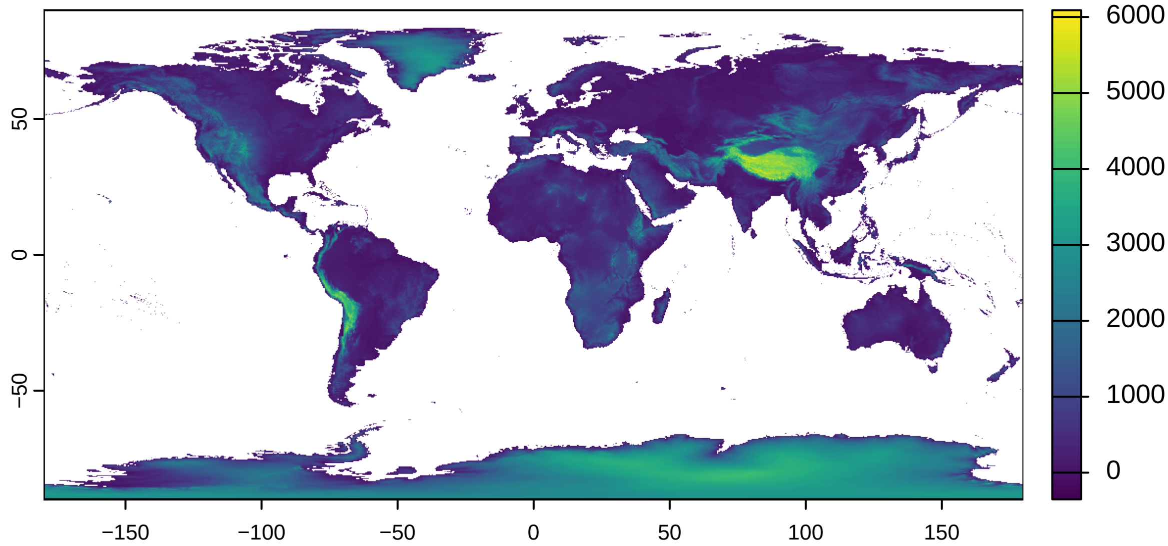
```r
terra::minmax(global_elevation_10min)
```

```
##     wc2.1_10m_elev
## min           -352
## max           6251
```

Basic Plotting

Method 1: Quick plotting using `terra::plot()`:

```r
terra::plot(global_elevation_10min)
```



Adding titles to the basic plot:

```
# Use `title("Global Elevation (terra::plot)")`
```

Method 2: Plotting with `ggplot2::geom_raster()`:

Step 1: Load the required packages:

```
pacman::p_load(viridis)
```

Step 2: Convert SpatRaster to data frame for `ggplot2` using `terra::as.data.frame()`:

```
elevation_df <- terra::as.data.frame(
  global_elevation_10min,
  xy = TRUE # to include the longitude (x) and latitude (y)
)
```

Step 3: Inspect the data structure:

```
head(elevation_df)
```

```
##                x        y wc2.1_10m_elev
## 82927 -38.91667 83.58333              0
## 82928 -38.75000 83.58333              0
## 82929 -38.58333 83.58333              0
```

```
## 82930 -38.41667 83.58333              0
## 82931 -38.25000 83.58333              0
## 82934 -37.75000 83.58333              0
```

```
glimpse(elevation_df)
```

```
## Rows: 808,053
## Columns: 3
## $ x             <dbl> -38.91667, -38.75000, -38.58333, -38.41667, -38.25000, ~
## $ y             <dbl> 83.58333, 83.58333, 83.58333, 83.58333, 83.58333, 83.58~
## $ wc2.1_10m_elev <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0~
```

```
colnames(elevation_df)
```

```
## [1] "x"              "y"              "wc2.1_10m_elev"
```

Step 4: Rename the the value columm for easier plotting:

```
# Extract the original name first:
value_col_name <- names(elevation_df)[3]
value_col_name
```

```
## [1] "wc2.1_10m_elev"
```

```
elevation_df <- elevation_df |>
  rename(elevation = dplyr::all_of(value_col_name)
) |>
  filter(!is.na(elevation))

colnames(elevation_df)
```

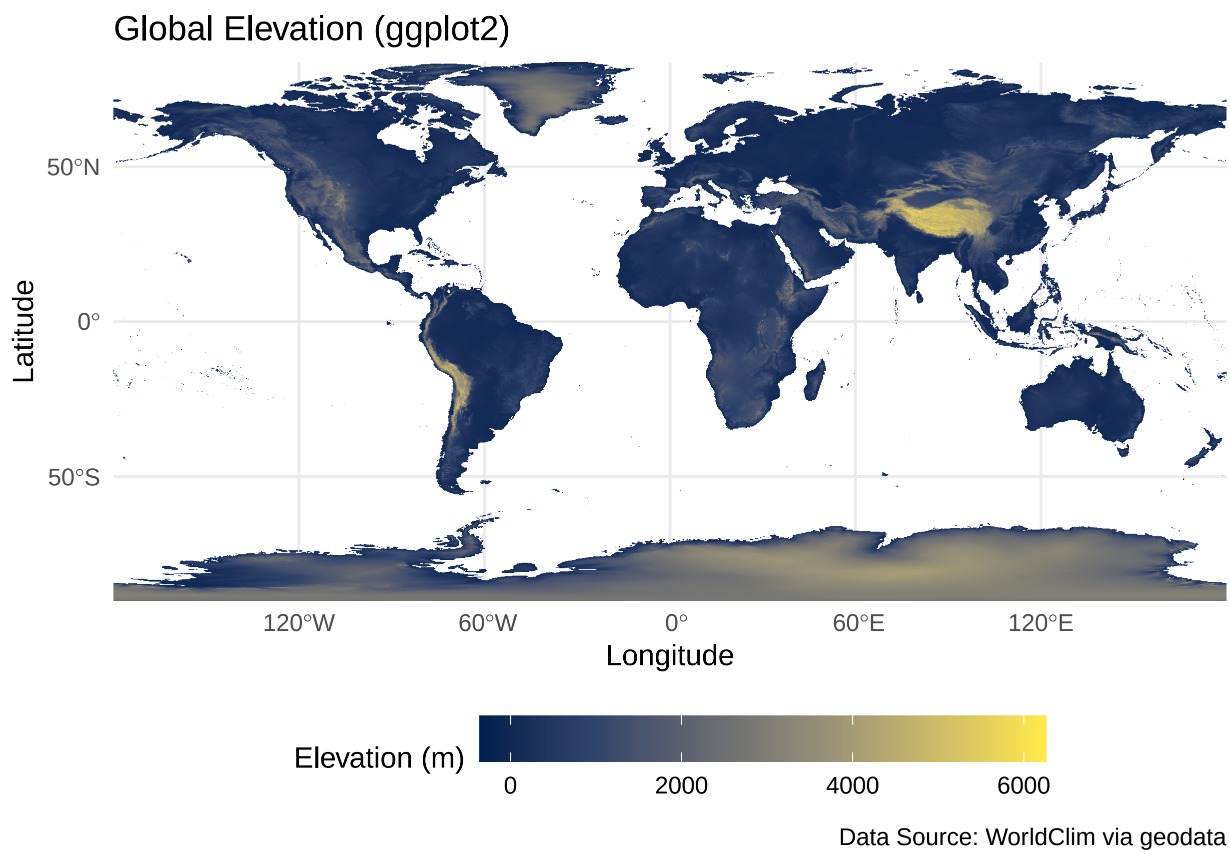
```
## [1] "x"         "y"         "elevation"
```

Step 5: Create the plot in `ggplot`:

```
elevation_df |> ggplot() +
  geom_raster(aes(x = x, y = y, fill = elevation)) +
  # Add a suitable color scale
  scale_fill_viridis_c(
    option = "cividis", name = "Elevation (m)"
  ) +
  labs(
    title = "Global Elevation (ggplot2)",
    x = "Longitude", y = "Latitude",
    caption = "Data Source: WorldClim via geodata"
  ) +
  # Use coord_sf() to ensure proper map aspect ratio
  coord_sf(
    crs = 4326,
    expand = FALSE # to remove the padding
```

```
  ) +
  theme_minimal() +
  theme(
    legend.position = "bottom",
    legend.key.width = unit(1.5, "cm") # make color bar wider
  )
```

```
## Warning: Raster pixels are placed at uneven horizontal intervals and will be shifted
## i Consider using 'geom_tile()' instead.
```



Simple Raster Operations: Cropping using `terra::crop()`:

Step 1: Load the required packages:

```
pacman::p_load(sf, naturalearth)
```

```
## Installing package into '/home/cavemancoder/R/x86_64-pc-linux-gnu-library/4.5'
## (as 'lib' is unspecified)

## Warning: package 'naturalearth' is not available for this version of R
##
## A version of this package for your version of R might be available elsewhere,
## see the ideas at
## https://cran.r-project.org/doc/manuals/r-patched/R-admin.html#Installing-packages
```

```
## Warning: 'BiocManager' not available.  Could not check Bioconductor.
##
## Please use `install.packages('BiocManager')` and then retry.

## Warning in p_install(package, character.only = TRUE, ...):

## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'naturalearth'

## Warning in pacman::p_load(sf, naturalearth): Failed to install/load:
## naturalearth
```

Step 2: Ensure that global elevation raster exists.

Step 3: Get target boundary polygon from `rnaturalearth`. For this example, the target boundary is South America.

```r
sa_boundary_sf <- rnaturalearth::ne_countries(
  scale = "medium",
  continent = "South America",
  returnclass = "sf"
) |>
  select(
    iso_a3 = adm0_a3, name, geometry
  )

sa_boundary_sf
```

```
## Simple feature collection with 13 features and 2 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -109.4341 ymin: -55.8917 xmax: -34.80547 ymax: 12.43437
## Geodetic CRS:  WGS 84
## First 10 features:
##     iso_a3          name                       geometry
## 5      VEN    Venezuela MULTIPOLYGON (((-60.82119 9...
## 9      URY      Uruguay MULTIPOLYGON (((-53.37061 -...
## 23     FLK Falkland Is. MULTIPOLYGON (((-58.8502 -5...
## 51     SUR     Suriname MULTIPOLYGON (((-54.15596 5...
## 82     PER         Peru MULTIPOLYGON (((-69.96592 -...
## 83     PRY     Paraguay MULTIPOLYGON (((-58.15977 -...
## 150    GUY       Guyana MULTIPOLYGON (((-60.74214 5...
## 178    ECU      Ecuador MULTIPOLYGON (((-75.28447 -...
## 195    COL     Colombia MULTIPOLYGON (((-71.31973 1...
## 199    CHL        Chile MULTIPOLYGON (((-109.28 -27...
```

Step 4: Ensure that the CRS in the raster and vector polygons are matching:

```r
raster_crs_str <- terra::crs(global_elevation_10min, proj = TRUE)
raster_crs_str2 <- terra::crs(global_elevation_10min, describe = TRUE)
vector_crs_sf <- sf::st_crs(sa_boundary_sf)

paste("Raster CRS:", raster_crs_str)
```

```
## [1] "Raster CRS: +proj=longlat +datum=WGS84 +no_defs"
```

```r
paste("Raster CRS:", raster_crs_str2$name)
```

```
## [1] "Raster CRS: WGS 84"
```

```r
paste("Vector CRS:", vector_crs_sf$Name)
```

```
## [1] "Vector CRS: WGS 84"
```

Step 5: Make the necessary transformations:

```r
# Transform the vector boundary to match the raster's CRS
sa_boundary_sf <- sa_boundary_sf |>
  sf::st_transform(
    crs = terra::crs(global_elevation_10min)
  )
```

Step 6: Crop the raster using the sf polygon boundary using `terra::crop()`:

```r
sa_elevation <- terra::crop(
  global_elevation_10min, sa_boundary_sf,
  mask = TRUE # to make pixels outside the polygon equal to NA
)
```

```
## Warning in x@pntr$crop_mask(y@pntr, snap[1], touches[1], extend[1], opt): GDAL
## Message 1: DeprecationWarning: 'Memory' driver is deprecated since GDAL 3.11.
## Use 'MEM' onwards. Further messages of this type will be suppressed.
```

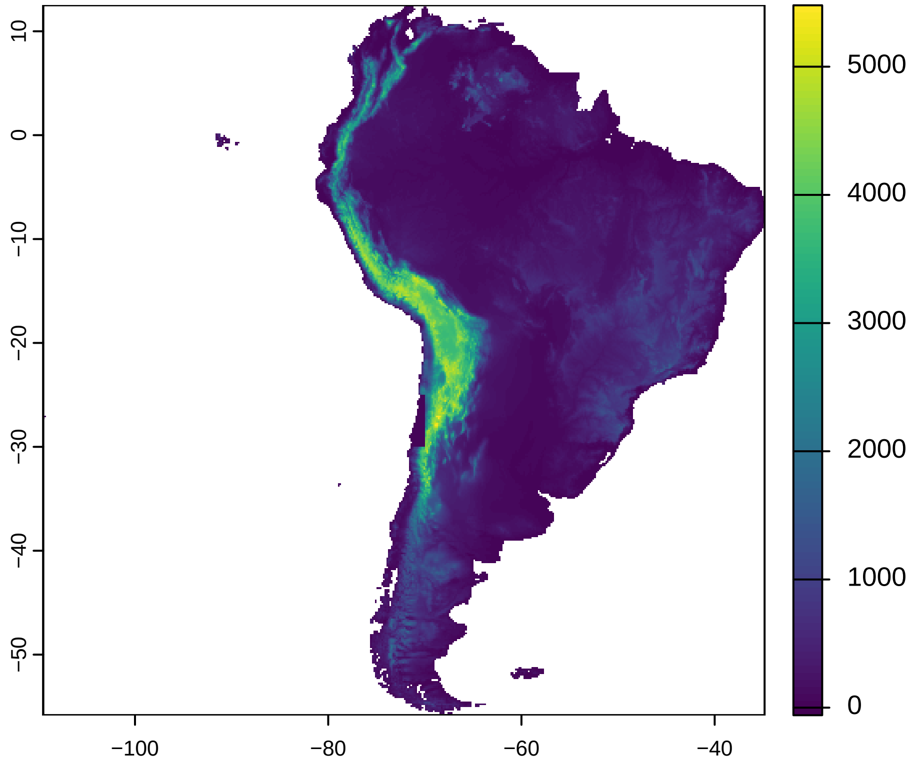
Step 7: Inspect the cropped raster:

```r
sa_elevation
```

```
## class       : SpatRaster
## size        : 410, 448, 1  (nrow, ncol, nlyr)
## resolution  : 0.1666667, 0.1666667  (x, y)
## extent      : -109.5, -34.83333, -55.83333, 12.5  (xmin, xmax, ymin, ymax)
## coord. ref. : lon/lat WGS 84 (EPSG:4326)
## source(s)   : memory
## varname     : wc2.1_10m_elev
## name        : wc2.1_10m_elev
## min value   :            -59
## max value   :           5479
```

Step 8: Plot the cropped raster using `terra::plot()`:

```r
terra::plot(sa_elevation)
```



Project Exercise: Explore Your Region's Elevation

Goal: Create a basic elevation map for a chosen country using `terra` and `ggplot2`.

Step 1: Choose Country and Setup:

```
pacman::p_load(sf, tidyverse, terra, geodata, giscoR, ggplot2, viridis)
chosen_country_name = "Philippines"

country_code_ph <- country_codes(chosen_country_name)$ISO3
country_code_ph
```

```
## [1] "PHL"
```

Step 2: Downlaod Elevation Data:

```
my_country_elev <- geodata::elevation_30s(
  country = country_code_ph,
  path = tempdir()
)
```

Step 3: Inspect.

```
my_country_elev
```

```
## class       : SpatRaster
## size        : 2016, 1200, 1  (nrow, ncol, nlyr)
## resolution  : 0.008333333, 0.008333333  (x, y)
## extent      : 116.8, 126.8, 4.4, 21.2  (xmin, xmax, ymin, ymax)
## coord. ref. : lon/lat WGS 84 (EPSG:4326)
## source      : PHL_elv_msk.tif
## name        : PHL_elv_msk
## min value   :         -67
## max value   :        2804
```

```
terra::crs(my_country_elev)
```

```
## [1] "GEOGCRS[\"WGS 84\",\n    ENSEMBLE[\"World Geodetic System 1984 ensemble\",\n        MEMBER[\"Wo
```

Step 4: Map with `ggplot`.

First, convert raster file to data frame:

```
# Convert raster file to data frame
my_country_df <- as.data.frame(my_country_elev, xy = TRUE)

my_country_df <- my_country_df |>
  rename(
    elevation = names(my_country_df)[3]
  ) |>
  filter(!is.na(names(my_country_df)[1]))

head(my_country_df)
```

```
##          x        y elevation
## 1 121.9208 20.92917         5
## 2 121.9042 20.90417         5
## 3 121.8458 20.82917         7
## 4 121.8542 20.82917       179
## 5 121.8625 20.82917       165
## 6 121.8375 20.82083       162
```

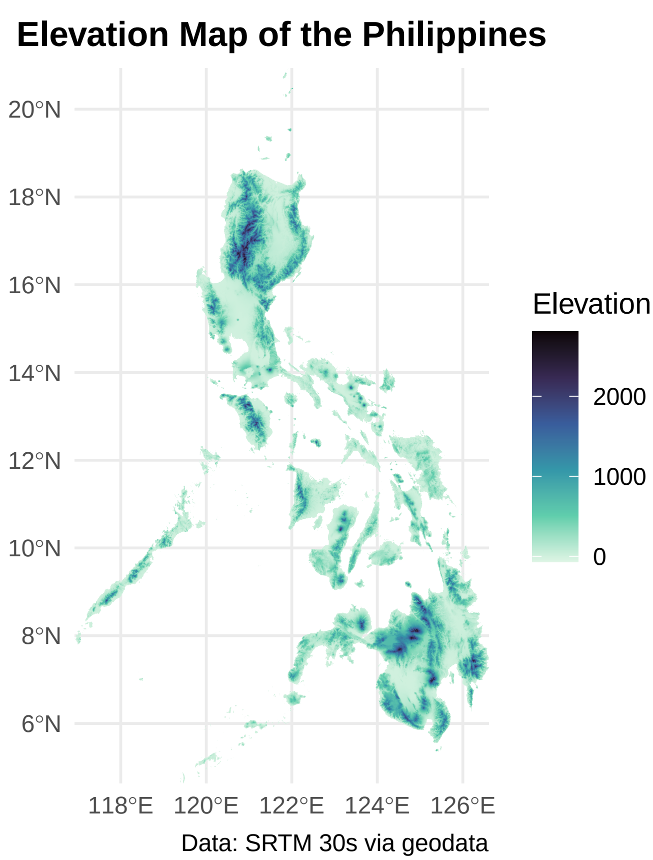
Plot with `ggplot2`:

```
my_country_map <- my_country_df |> ggplot() +
  geom_raster(
    aes(x = x, y = y, fill = elevation)
  ) +
  scale_fill_viridis_c(
    option = "mako",
    name = "Elevation",
    direction = -1
  ) +
  labs(
```

```r
    title = paste("Elevation Map of the", chosen_country_name),
    caption = "Data: SRTM 30s via geodata"
  ) +
  coord_sf(
    crs = terra::crs(my_country_elev),
    expand = FALSE # to remove the padding
  ) +
  theme_minimal() +
  theme(
    axis.title = element_blank(),
    plot.title = element_text(hjust = 0.5, face = "bold")
  )

my_country_map
```

```
## Warning: Raster pixels are placed at uneven horizontal intervals and will be shifted
## i Consider using 'geom_tile()' instead.
## Raster pixels are placed at uneven horizontal intervals and will be shifted
## i Consider using 'geom_tile()' instead.
```




```r
# saving
ggsave(
  "my_country_elevation_map.png", my_country_map,
  width = 8, height = 6,
```

```r
  dpi = 600, bg = "white"
)
```

```
## Warning: Raster pixels are placed at uneven horizontal intervals and will be shifted
## i Consider using `geom_tile()` instead.
## Raster pixels are placed at uneven horizontal intervals and will be shifted
## i Consider using `geom_tile()` instead.
```