Chapter 8

The Caveman Coder

2025-11-03

Hands-On: Visualizing Elevation Data

Setting up the environment

Step 1: Load necessary packages:

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

Step 2: Set general theme:

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

Getting Elevation Data for Switzerland

Step 1: Define country code:

```
country_iso <- "CHE"
```

Step 2: Download/load elevation data using `geodata::elevation_30s`.

```
swiss_elev_raster <- geodata::elevation_30s(
  country = country_iso,
  path = tempdir()
)
```

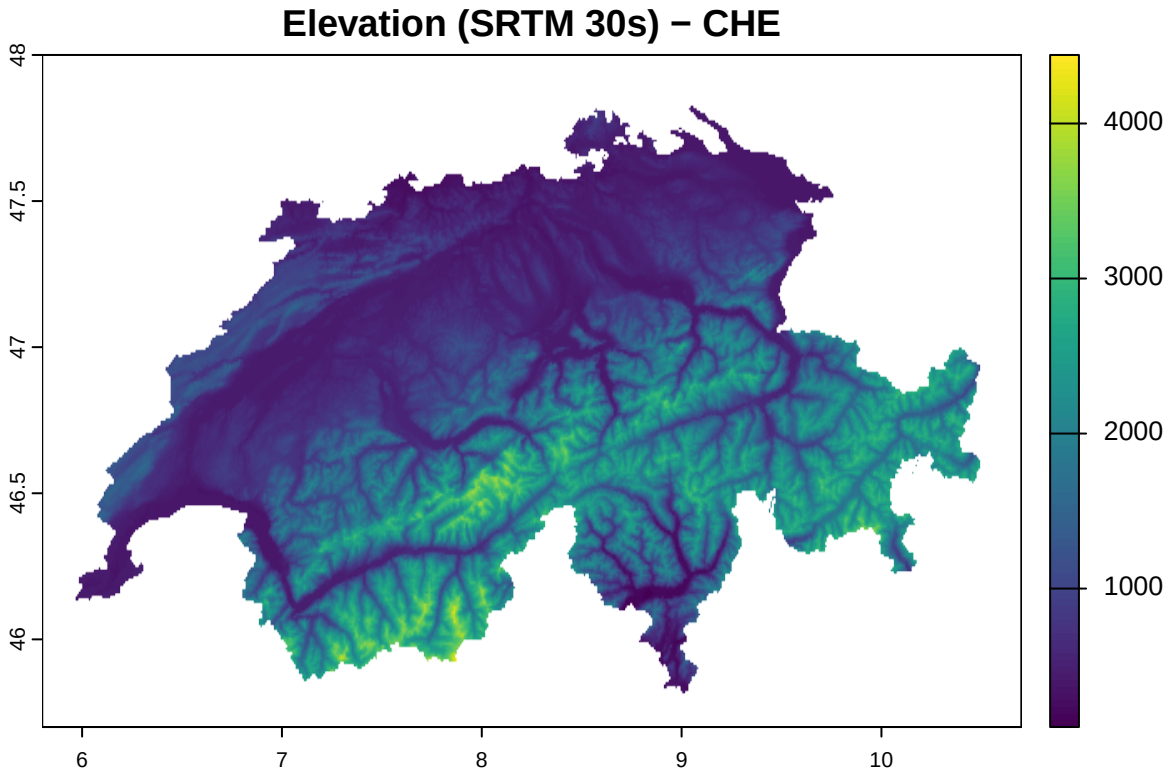
Step 3: Inspect the loaded raster data:

```
swiss_elev_raster
```

```
## class       : SpatRaster
## size        : 276, 588, 1  (nrow, ncol, nlyr)
## resolution  : 0.008333333, 0.008333333  (x, y)
## extent      : 5.8, 10.7, 45.7, 48  (xmin, xmax, ymin, ymax)
## coord. ref. : lon/lat WGS 84 (EPSG:4326)
## source      : CHE_elv_msk.tif
## name        : CHE_elv_msk
## min value   :         107
## max value   :        4442
```

Step 4: Generate a quick plot using `terra::plot()`:

```
terra::plot(
  swiss_elev_raster,
  main = paste("Elevation (SRTM 30s) -", country_iso)
)
```



Calculate Terrain Attributes (Slope, Aspect, Hillshade)

Step 1: Calculate Slope and Aspect using `terra::terrain()`:

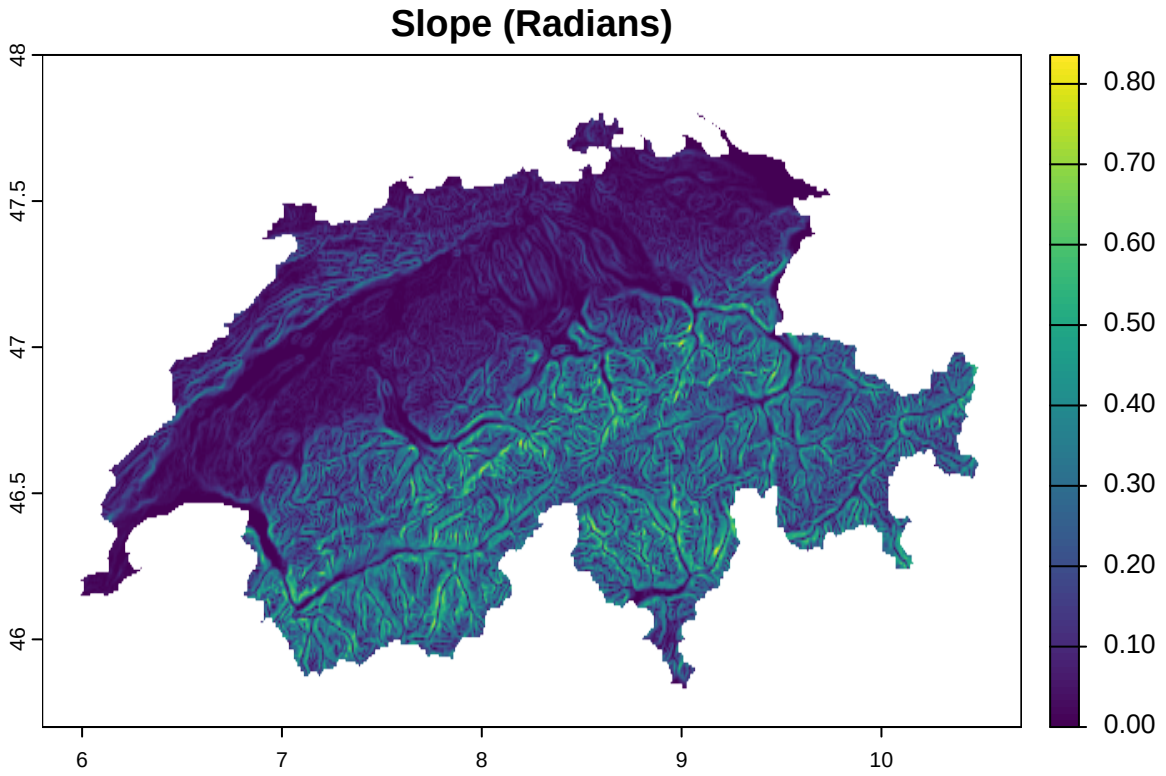
```
terrain_attributes <- terra::terrain(
  swiss_elev_raster,
  v = c("slope", "aspect"),
  unit = "radians"
)
terrain_attributes
```

```
## class       : SpatRaster
## size        : 276, 588, 2  (nrow, ncol, nlyr)
## resolution  : 0.008333333, 0.008333333  (x, y)
## extent      : 5.8, 10.7, 45.7, 48  (xmin, xmax, ymin, ymax)
## coord. ref. : lon/lat WGS 84 (EPSG:4326)
## source(s)   : memory
## names       :     slope,   aspect
## min values  : 0.0000000, 0.000000
## max values  : 0.8359931, 6.283185
```

Step 2: Plot Slope and Aspect individually with MULTIPLE layers:

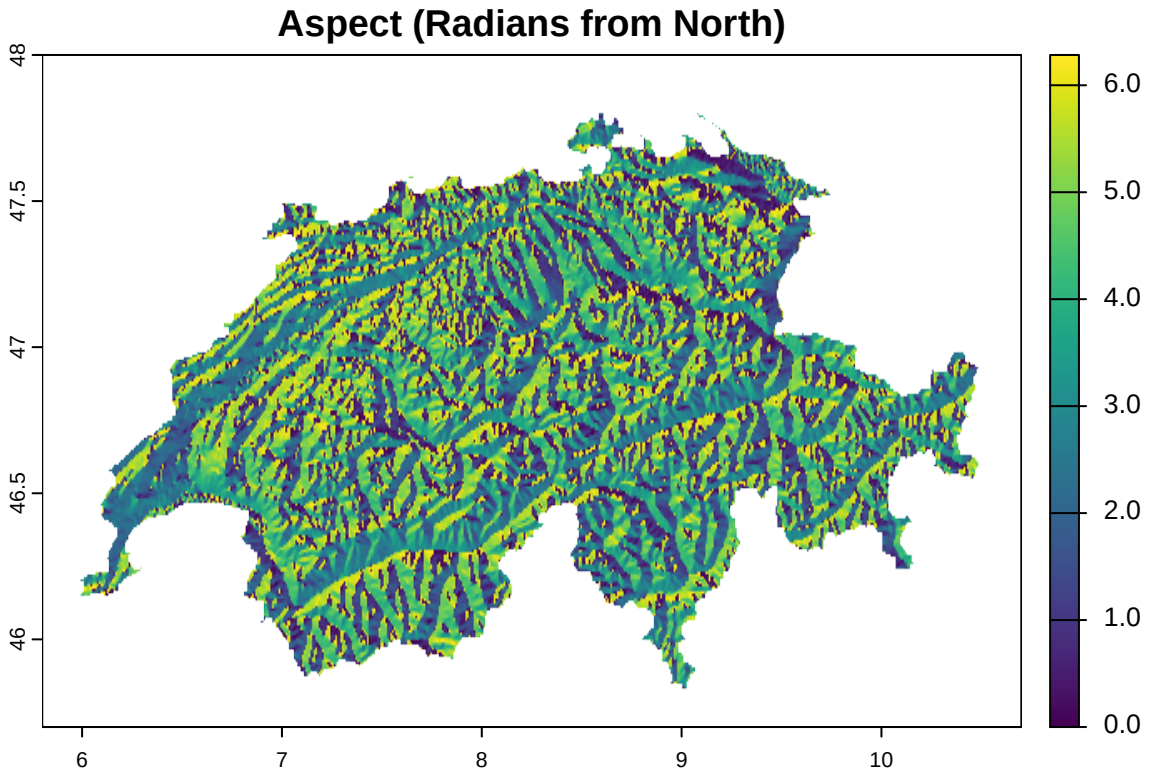
Plotting `Slope`:

```
terra::plot(
  terrain_attributes$slope,
  main = "Slope (Radians)"
)
```



Plotting `Aspect`:

```
terra::plot(
  terrain_attributes$aspect,
  main = "Aspect (Radians from North)",
  col = viridis(100)
)
```



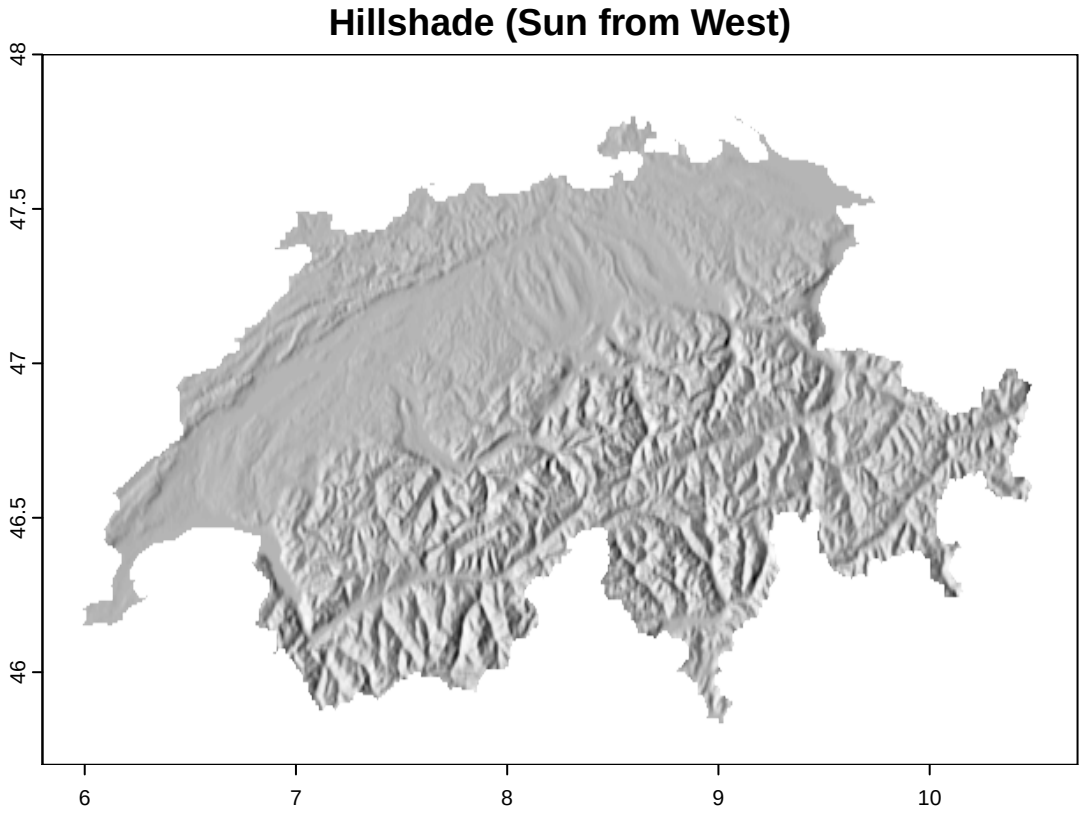
Step 3: Calculate Hillshade using `terra::shade()`:

```
hillshade <- terra::shade(
  terrain_attributes$slope,
  terrain_attributes$aspect,
  angle = 45, # degrees above the horizon
  direction = 270 # setting for the sun's direction (270 = West, 0 = North, etc.)
)

hillshade
```

```
## class       : SpatRaster
## size        : 276, 588, 1  (nrow, ncol, nlyr)
## resolution  : 0.008333333, 0.008333333  (x, y)
## extent      : 5.8, 10.7, 45.7, 48  (xmin, xmax, ymin, ymax)
## coord. ref. : lon/lat WGS 84 (EPSG:4326)
## source(s)   : memory
## name        :  hillshade
## min value   : 0.01180136
## max value   : 0.99744209
```

Step 4: Plot the Hillshade (standard is grayscale)

```
terra::plot(
  hillshade,
  col = gray(0:100 / 100), # Creates 101 shades of gray
  legend = FALSE, # Hides color legend
  main = "Hillshade (Sun from West)"
)
```



Creating Shaded Relief Maps with `ggplot2` (The Cool Part!)

Step 1: Load necessary packages and set the environment

```
pacman::p_load(terra, tidyverse, dplyr, tidyterra, ggnewscale)
```

Rename layers for clarity:

```
names(swiss_elev_raster) <- "elevation"
names(hillshade) <- "hillshade_val"
```

Convert `swiss_elev_raster` to a data frame:

```
elev_df_gg <- terra::as.data.frame(
  swiss_elev_raster,
  xy = TRUE,
  na.rm = TRUE
)

hillshade_df_gg <- terra::as.data.frame(
  hillshade,
  xy = TRUE,
  na.rm = TRUE
)

# Define map limits
limits <- terra::minmax(swiss_elev_raster)
```

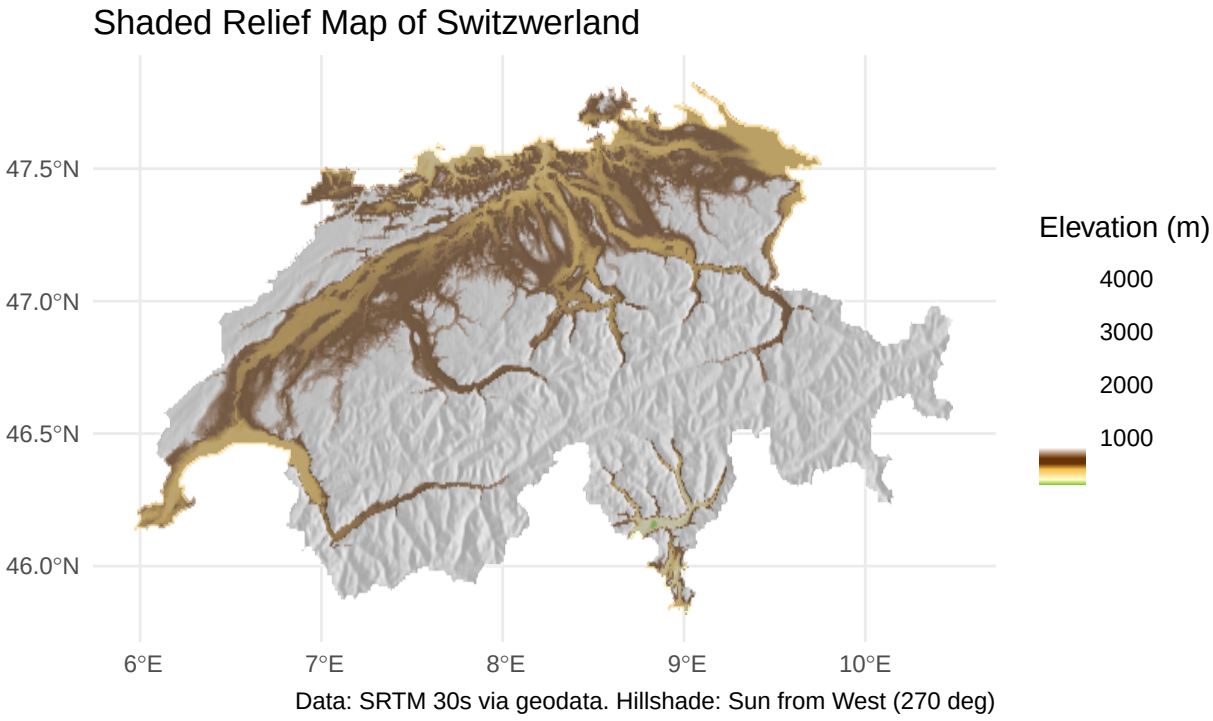
Step 2: Create the ggplot map with hillshade overlay:

```
shaded_relief_map <- ggplot() +
  geom_raster(
    data = hillshade_df_gg,
    aes(x = x, y = y, fill = hillshade_val),
    show.legend = FALSE
  ) +
  # Change default color palette
  scale_fill_gradientn(
    colors = hcl.colors(12, "Light Grays", rev = TRUE),
    na.value = NA # Transparent for any missing cells
  ) +
  # Rendering the elevation raster on top of the hillshade
  ggnewscale::new_scale_fill() +
  geom_raster(
    data = elev_df_gg,
    aes(x = x, y = y, fill = elevation),
    alpha = 0.5
  ) +
  # Choose a continuous hypso palette for elevation
  tidyterra::scale_fill_hypso_tint_c(
    palette = "dem_print",
    limits = limits
  ) +
  # Labels and theme
  labs(
    title = "Shaded Relief Map of Switzwerland",
    fill = "Elevation (m)",
    caption = paste(
      "Data: SRTM 30s via geodata.",
      "Hillshade: Sun from West (270 deg)"
    )
  ) +
  # Ensure correct aspect ratio for maps
  coord_sf(crs = terra::crs(swiss_elev_raster)) +
  theme_minimal() +
  theme(
    legend.position = "right",
    axis.title = element_blank()
  )
```

```
## Warning in (x - from[1])/diff(from): Recycling array of length 1 in vector-array arithmetic is depre
##   Use c() or as.vector() instead.
```

```
shaded_relief_map
```

```
## Warning in (x - from[1])/diff(from): Recycling array of length 1 in vector-array arithmetic is depre
##   Use c() or as.vector() instead.
## Warning in (x - from[1])/diff(from): Recycling array of length 1 in vector-array arithmetic is depre
##   Use c() or as.vector() instead.
## Warning in (x - from[1])/diff(from): Recycling array of length 1 in vector-array arithmetic is depre
##   Use c() or as.vector() instead.
```



Generation and Styling

Step 1: Load necessary packages

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

Step 2: Generate contour lines using `terra::as.contour()`:

```
contour_lines_terra <- terra::as.contour(
  swiss_elev_raster, nlevels = 10
)
```

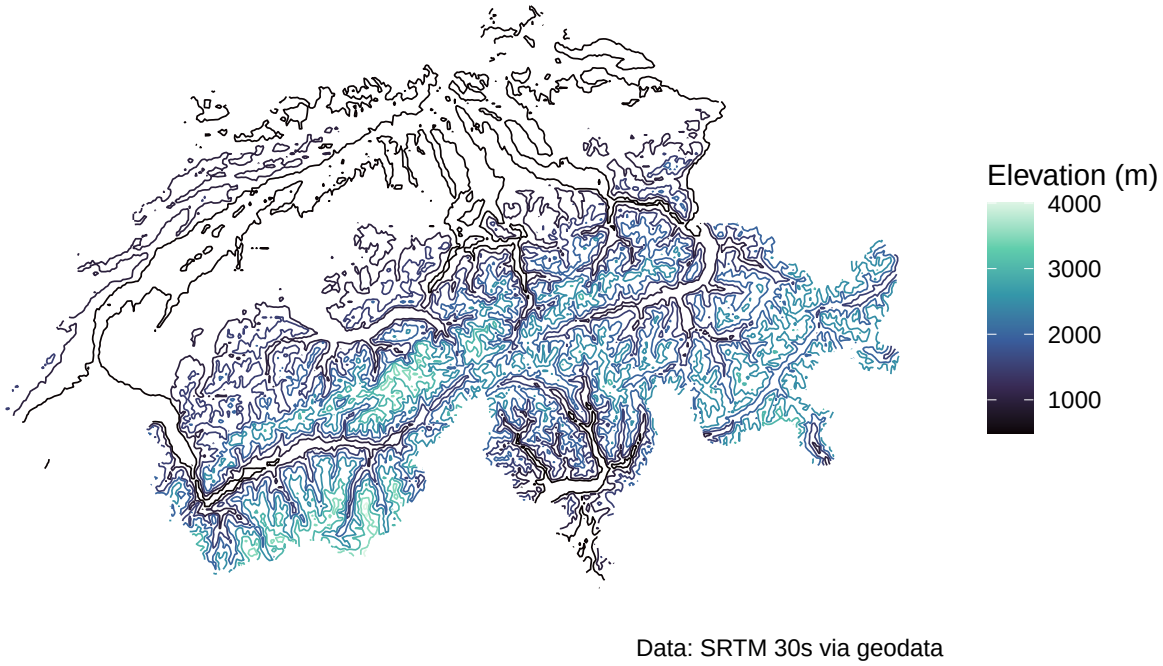
Step 3: Convert terra's contour output (SpatRaster) to an `sf` object:

```
contour_lines_sf <- contour_lines_terra |>
  sf::st_as_sf()

contour_lines_sf
```

```
## Simple feature collection with 8 features and 1 field
## Geometry type: MULTILINESTRING
## Dimension:     XY
## Bounding box:  xmin: 6.090332 ymin: 45.82315 xmax: 10.48258 ymax: 47.78339
## Geodetic CRS:  WGS 84
##   level                       geometry
## 1   500 MULTILINESTRING ((7.483525 ...
## 2  1000 MULTILINESTRING ((6.338781 ...
## 3  1500 MULTILINESTRING ((6.091667 ...
## 4  2000 MULTILINESTRING ((6.7951 46...
## 5  2500 MULTILINESTRING ((6.902618 ...
## 6  3000 MULTILINESTRING ((7.004883 ...
## 7  3500 MULTILINESTRING ((7.247024 ...
## 8  4000 MULTILINESTRING ((7.857842 ...
```

Step 4: Map contours with `ggplot2`:

```
contour_map <- ggplot() +
  # Layer 1: Plot of contour lines using sf with the 'level' column mapped to
  # color aesthetic
  geom_sf(
    data = contour_lines_sf,
    aes(color = level),
    linewidth = 0.3
  ) +
  # Use a sequential color scale like scale_color_viridis_c
  scale_color_viridis_c(
    option = "mako",
    name = "Elevation (m)"
  ) +
  # Labels and theme
  labs(
    title = "Contour Map of Switzerland",
    caption = "Data: SRTM 30s via geodata"
  ) +
  # Themes
  theme_minimal() +
  theme(
    axis.text = element_blank(),
    panel.grid = element_blank()
  )

contour_map
```



Combining Elevation with Other Features

Example of an elevation map including the `geom_sf` layer for boundary

Step 1: Load the necessary packages

```
pacman::p_load(terra, tidyverse, tidyterra, ggnewscale, geodata)
```

Step 2: Get country boundary layer and add to the shaded relief map earlier

```
ch_boundary_sf <- geodata::gadm(
  country = "CHE", level = 0, path = tempdir()
) |>
  sf::st_as_sf() |>
  sf::st_transform(
    crs = terra::crs(swiss_elev_raster)
  )
```

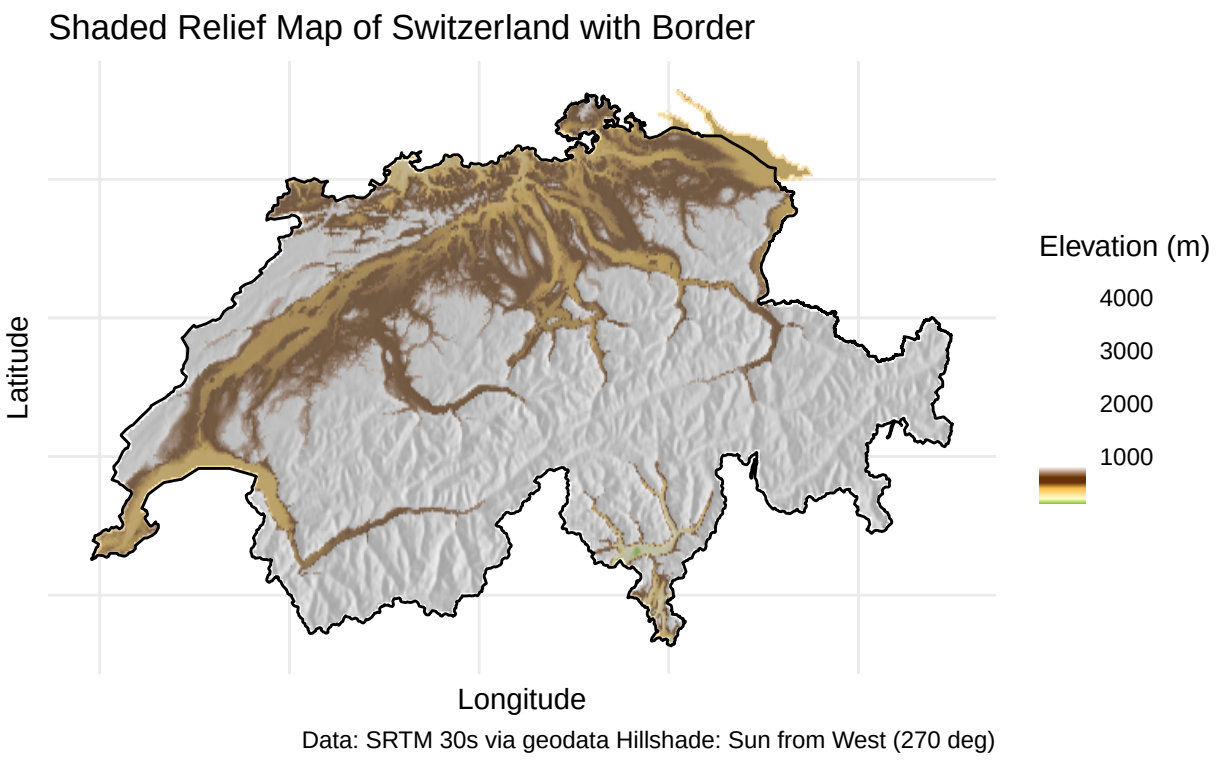
Add to relief map:

```
shaded_relief_map_with_border <- shaded_relief_map +
  # Add boundary layer
  geom_sf(
    data = ch_boundary_sf,
    fill = NA,
    color = "black",
    linewidth = 0.5
  ) +
  labs(
    title = "Shaded Relief Map of Switzerland with Border",
    x = "Longitude",
    y = "Latitude",
    caption = paste(
      "Data: SRTM 30s via geodata",
      "Hillshade: Sun from West (270 deg)"
    )
  ) +
  coord_sf(crs = terra::crs(swiss_elev_raster)) +
  theme_minimal() +
  theme(
    legend.position = "right",
    axis.text = element_blank()
  )
```

```
## Coordinate system already present. Adding new coordinate system, which will
## replace the existing one.
```

```
shaded_relief_map_with_border
```

```
## Warning in (x - from[1])/diff(from): Recycling array of length 1 in vector-array arithmetic is depre
##   Use c() or as.vector() instead.
## Warning in (x - from[1])/diff(from): Recycling array of length 1 in vector-array arithmetic is depre
##   Use c() or as.vector() instead.
## Warning in (x - from[1])/diff(from): Recycling array of length 1 in vector-array arithmetic is depre
##   Use c() or as.vector() instead.
```



Project Exercise: Create a Topographic Map with Styled Elevation

Step 1: Choose region

```
country_iso <- "PHL" # Philippines
```

Step 2: Get data

```
elevation_raster <- geodata::elevation_30s(
  country = country_iso, path = tempdir()
)

boundary_sf <- geodata::gadm(
  country = country_iso, level = 0, path = tempdir()
) |>
  sf::st_as_sf() |>
  sf::st_transform(
    crs = terra::crs(elevation_raster)
  )
```

Step 3: Calculate Terrain Attributes

```
terrain_attrs <- terra::terrain(
  elevation_raster,
  v = c("slope", "aspect"),
  unit = "degrees"
)

hillshade_raster <- terra::shade(
  terrain_attrs$slope,
  terrain_attrs$aspect,
  angle = 45,
```

```
  direction = 225 # Sun from NE
)

names(hillshade_raster) <- "hillshade_val"
names(elevation_raster) <- "elevation"
```

Step 4: Prepare for plotting

```
elev_df <- terra::as.data.frame(
  elevation_raster,
  xy = TRUE,
  na.rm = TRUE
)

hill_df <- terra::as.data.frame(
  hillshade_raster,
  xy = TRUE,
  na.rm = TRUE
)
```

Step 5: Set the limits

```
limits <- terra::minmax(elevation_raster)
```

Step 5: Plot!

```
topo_map_project <- ggplot() +
  # Hillshade layer
  geom_raster(
    data = hill_df,
    aes(x = x, y = y, fill = hillshade_val),
    show.legend = FALSE
  ) +
  scale_fill_gradientn(
    colors = hcl.colors(
      12, "Light Grays", rev = TRUE
    ),
    na.value = NA
  ) +
  # Elevation layer
  ggnewscale::new_scale_fill() +
  geom_raster(
    data = elev_df,
    aes(x = x, y = y, fill = elevation)
  ) +
  tidyterra::scale_fill_hypso_tint_c(
    palette = "dem_poster",
    limits = limits,
    alpha = 0.5
  ) +
  # Optional boundary
  geom_sf(
    data = boundary_sf,
    fill = NA,
    color = "black",
    linewidth = 0.1
```

```
  ) +
  # Labels
  labs(
    title = "Shaded Relief Map of the Philippines",
    caption = "Data: SRTM 30s via geodata"
  ) +
  geom_sf(
    crs = terra::crs(elevation_raster), expand = FALSE
  ) +
  theme_void() +
  theme(
    legend.position = "right",
    plot.title = element_text(hjust = 0.5, face = "bold")
  )
```

```
## Warning in (x - from[1])/diff(from): Recycling array of length 1 in vector-array arithmetic is depre
##   Use c() or as.vector() instead.
```

```
## Warning in layer_sf(geom = GeomSf, data = data, mapping = mapping, stat = stat,
## : Ignoring unknown parameters: `crs` and `expand`
```

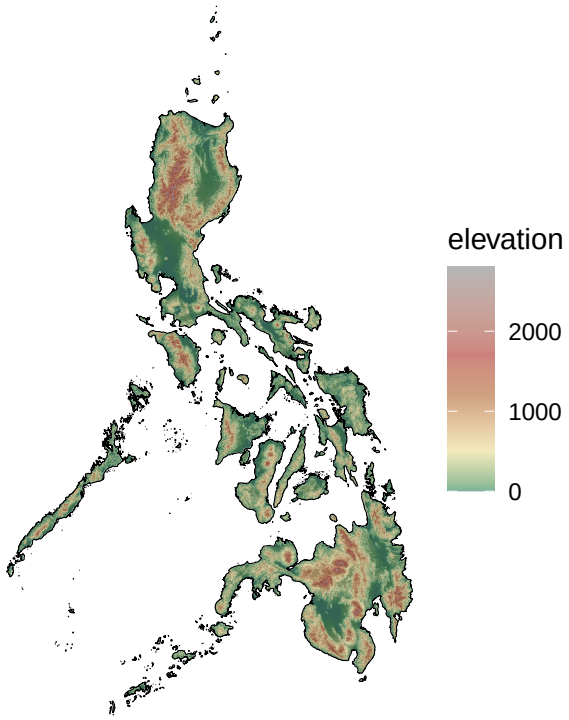
```
topo_map_project
```

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

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

```
## Warning in (x - from[1])/diff(from): Recycling array of length 1 in vector-array arithmetic is depre
##   Use c() or as.vector() instead.
## Warning in (x - from[1])/diff(from): Recycling array of length 1 in vector-array arithmetic is depre
##   Use c() or as.vector() instead.
## Warning in (x - from[1])/diff(from): Recycling array of length 1 in vector-array arithmetic is depre
##   Use c() or as.vector() instead.
```

Shaded Relief Map of the Philippines



Data: SRTM 30s via geodata

Step 7: Saving

```
# ggsave("topo_map_phl.png", topo_map_project, width = 7, height = 8, dpi = 300, bg = "white")
```