````
---
title: "Anomalías de los volumenes en embalses de España"
subtitle: "Actividad Guiada 2"
author:
  - name: Rubén Garrido Hidalgo
    email: rubengh2002@gmail.com
format: html
editor: source
---

## Hillshade

## Librerías necesarias
```{r echo=FALSE,message=FALSE,error=FALSE,warning=FALSE}
library(knitr)

tb <- data.frame(paquete=c("tidyverse", "sf", "elevatr", "terra",
"whitebox",
                           "tidyterra", "giscoR", "ggnewscale", "ggblend"),
                 descripcion=c("Conjunto de paquetes (visualización y
manipulación de datos): ggplot2, dplyr, purrr,etc.", "Simple Feature:
importar, exportar y manipular datos vectoriales",
                               "Acceso a datos de elevación desde varias
API",
                           "Importar, exportar y manipular raster (paquete
sucesor de raster)",
                           "Una interfaz R para la biblioteca
'WhiteboxTools', que es una plataforma avanzada de análisis de datos
geoespaciales",
                           "Funciones auxilares para trabajar con {terra}",
                           "Límites administrativos del mundo",
                           "Extensión para ggplot2 de múltiples 'scales'",
"Extensión para mezclar colores de gráficos ggplot"))

kable(tb,booktabs = TRUE, col.names=c("Paquete","Descripción"))
```

```{r,message=FALSE,error=FALSE,warning=FALSE}

if (!requireNamespace("elevatr", quietly = TRUE))
install.packages("elevatr")
if (!requireNamespace("terra", quietly = TRUE)) install.packages("terra")
if (!requireNamespace("ggnewscale", quietly = TRUE))
install.packages("ggnewscale")
if (!requireNamespace("tidyterra", quietly = TRUE))
install.packages("tidyterra")
if (!requireNamespace("units", quietly = TRUE)) install.packages("units")
if (!requireNamespace("ggblend", quietly = TRUE))
install.packages("ggblend")
if (!requireNamespace("giscoR", quietly = TRUE)) install.packages("giscoR")

library(sf)
library(elevatr)
library(tidyverse)
library(terra)
library(ggnewscale)
library(tidyterra)
library(giscoR)
````
````

```
library(units)
library(ggblend)

```

Comenzamos con la extracción y representación de los datos, en este caso nuestro área de interés será España, obtenemos los limites administrativos con la librería giscoR y la función gisco_get_countries()

```{r}
esp <- gisco_get_countries(country = "Spain", resolution = "03")

plot(esp)
```

El CNIG es la instutución que gestiona los datos cartográficos de España, por ello vamos a descargar de ahí la capa hidrográfica para obtener la información de los lagos.

```{r}
#Importamos los datos
esp_lagos <- st_read("./SIANE_CARTO_BASE_S_10M/anual/20150101/
se89_10_hidro_demt_a_x.shp")
str(esp_lagos)

#Filtramos por los lagos más garndes
lagos_grandes <-esp_lagos %>%
  mutate(areakm = set_units(st_area(geometry), "m^2") |>
                   set_units("km^2")) %>%
  filter(areakm > set_units(50, "km^2"))

plot(lagos_grandes)

```

Modelo digital de terreno (MDT)

La función `get_elev_raster()` nos permite descargar un MDT de cualquier región del mundo a través de diferentes proveedores en formato de ráster. Por defecto usa [AWS](https://registry.opendata.aws/terrain-tiles/).

Después de obtener el MDT de España debemos enmascarar los límites del país. La clase del objeto es *RasterLayer* del paquete `raster`, no obstante, el nuevo estándar es `terra` con la clase *SpatRaster*. Por eso lo convertimos y después aplicamos la máscara. Finalmente reproyectamos al sistema de coordenadas de España obtenido de los datos vectoriales.

```{r, warning=FALSE}
# Obtención mdt
mdt <- get_elev_raster(esp, z = 7)
mdt
plot(mdt)

#Cconvertimos a terra y enmascaramos la zona de interés
mdt <- rast(mdt) |>
         mask(vect(esp))

# Reproyectamos el mdt
```

```
mdt <- project(mdt, crs(lagos_grandes))

# Reproyectamos los limite de lagos
esp <- st_transform(esp, st_crs(lagos_grandes))
```

Antes de calcular el efecto de sombra, creamos un simple mapa de relieve. En `ggplot2` empleamos la geometría `geom_raster()` indicando la longitud, latitud y la variable para definir el color. Añadimos los límites de los lagos usando `geom_sf()` dado que se trata de un objeto *sf*. Aquí sólo indicamos el color de relleno con un azul claro. Con ayuda de `scale_fill_hypso_tint_c()` aplicamos una gama de colores correspondientes a un relieve, también llamado tintas hipsométricas, y definimos los cortes en la leyenda. En el resto de funciones hacemos ajustes de aspecto en la leyenda y en el estilo del gráfico.

```{r}
# Capa de elevación
names(mdt) <- "alt"

#Representación:
map <- ggplot() +
  geom_spatraster(data = mdt,
              aes(fill = alt)) +
  geom_sf(data = lagos_grandes,
          fill = "#c6dbef",
          colour = NA) +
  scale_fill_hypso_tint_c(breaks = c(180, 250, 500, 1000,
                                     1500,  2000, 2500,
                                     3000, 3500, 4000)) +
  guides(fill = guide_colorsteps()) +
  labs(fill = "m") +
  coord_sf() +
  theme_void() +
  theme(legend.position = "bottom",
        legend.key.height = unit(.5, "lines"),
        legend.key.width = unit(4, "lines"),
        legend.title.position = "right")
plot(map)
```

Internamente lo que hace `geom_spatraster()` es pasar el raster a un data.frame. `geom_raster()` sirve para una rejilla regular y `geom_tile()` si fuese irregular.

```{r}
df <- as.data.frame(mdt, xy = T)

dr_repre <- ggplot() +
  geom_raster(data = df,
              aes(x, y, fill = alt)) +
  theme_void()
plot(dr_repre)
```

Cálculo del efecto hillshade

Recordemos que el efecto hillshade es nada más que añadir una iluminación hipotética con respecto a una posición de una fuente de luz para así ganar profundidad. Las sombras dependen de dos variables, el acimut, el ángulo de la orientación sobra la superficie de una esfera, y la elevación, el ángulo de la altura de la fuente.

La información requerida para simular la iluminación es el modelo digital de terreno. La pendiente y la orientación podemos derivar del MDT usando la función `terrain()` del paquete `terra`. La unidad debe ser radianes. Una vez que tenemos todos los datos podemos hacer uso de la función `shade()` indicando el ángulo (elevación) y la dirección (acimut). El resultado es un ráster con valores entre 0 y 255, lo que nos indica sombras con bajos valores, siendo 0 negro y 255 blanco.

```{r, error=FALSE}
# Estimación de la pendiente
sl <- terrain(mdt, "slope", unit = "radians")
plot(sl)
# Estimación de la orientación
asp <- terrain(mdt, "aspect", unit = "radians")
plot(asp)

# Calculamos el efecto hillshade con 45° de elevación
hill_single <- shade(sl, asp,
      angle = 45,
      direction = 300,
      normalize= TRUE)


plot(hill_single, col = grey(1:100/100))
```


Combinar el relieve y el efecto de sombra

El problema para añadir al mismo tiempo el relieve con su tinta hipsométrica y el efecto hillshade dentro de `ggplot2` es que tenemos dos diferentes rellenos para cada capa. La solución consiste en usar la extensión `ggnewscale` que permite añadir múltiples *scales*. Primero añadimos con `geom_raster()` el hillshade, después definimos los tonos grises y antes de añadir la altitud incluimos la función `new_scale_fill()` para marcar otro relleno diferente. Para lograr el efecto es necesario dar un grado de transparencia a la capa del relieve, en este caso es del 70%. La elección de la dirección es importante, de ahí que debemos tener en cuenta siempre el lugar y el recorrido aparente del sol. ([sunearthtools](https://www.sunearthtools.com/dp/tools/pos_sun.php?lang=es)).

```{r, warning=FALSE}

names(hill_single)

# Representación
combination <- ggplot() +
  geom_spatraster(data = hill_single,
              aes(fill = hillshade),
              show.legend = FALSE) +
  scale_fill_distiller(palette = "Greys", na.value = NA) +
  new_scale_fill() +
```

```
  geom_spatraster(data = mdt,
              aes(fill = alt),
              alpha = .7) +
  scale_fill_hypso_tint_c(breaks = c(180, 250, 500, 1000,
                                     1500,  2000, 2500,
                                     3000, 3500, 4000)) +
  geom_sf(data = lagos_grandes,
          fill = "#c6dbef", colour = NA) +
  guides(fill = guide_colorsteps()) +
  labs(fill = "m") +
  coord_sf() +
  theme_void() +
    theme(legend.position = "bottom",
        legend.key.height = unit(.5, "lines"),
        legend.key.width = unit(4, "lines"),
        legend.title.position = "right")
plot(combination)
```

Sombras multidireccionales

Lo que hemos visto es un efecto unidireccional, aunque es lo más habitual, podemos crear un efecto más suave e incluso más realista combinando varias direcciones.

Simplemente mapeamos sobre un vector de varias direcciones al que se aplica la función `shade()` con una elevación fija. Después convertimos la lista de ráster en un objeto multidimensional de varias capas para reducirlas sumando todas las capas.

```{r}
# Pasamos varias direcciones a shade()
hillmulti <- map(c(270, 15, 60, 330), function(dir){
                    shade(sl, asp,
                          angle = 45,
                          direction = dir,
                          normalize= TRUE)}
  )

# Creamos un raster multidimensional y lo reducimos sumando
hillmulti <- rast(hillmulti) |> sum()

# Multidireccional
plot(hillmulti, col = grey(1:100/100))
# Unidireccional
plot(hill_single, col = grey(1:100/100))
```
Hacemos lo mismo como antes para visualizar el relieve con sombras multidireccionales.

```{r}
# convertimos el hillshade a xyz
names(hillmulti) <- "hillshade"

# Representación:
som_multi <- ggplot() +
  geom_spatraster(data = hillmulti,
              aes(fill = hillshade),
```

```
              show.legend = FALSE) +
  scale_fill_distiller(palette = "Greys", na.value = NA) +
  new_scale_fill() +
  geom_spatraster(data = mdt,
              aes(fill = alt),
              alpha = .7) +
  scale_fill_hypso_tint_c(breaks = c(180, 250, 500, 1000,
                                     1500,  2000, 2500,
                                     3000, 3500, 4000)) +
  geom_sf(data = lagos_grandes,
          fill = "#c6dbef", colour = NA) +
  guides(fill = guide_colorsteps()) +
  labs(fill = "m") +
  coord_sf() +
  theme_void() +
        theme(legend.position = "bottom",
        legend.key.height = unit(.5, "lines"),
        legend.key.width = unit(4, "lines"),
        legend.title.position = "right")
plot(som_multi)
```
Exportamos la representación aun archivo png:
```{r}
ggsave("som_multi.png", som_multi,
       height = 6, width = 12,
       units = "in",
       bg = "white")
```
La técnica de mezcla de colores es muy útil para obtener resultados notables en el efecto de sombreado. Desde hace poco el paquete `ggblend` ofrece esta posibilidad. Con el objetivo de combinar varias capas, es necesario insertar los objetos `geom_raster()` y los `scale_fill_*()` en una lista seperados por coma. Después le sigue el *pipe* con la función `blend("tipo_de_mezcal")` al que le sumamos los otros objetos de `ggplot2`. En este caso aplicamos la multiplicación como forma de mezcla.

```{r, eval=FALSE}
# Representación:
m <- ggplot() +
  list(
  geom_spatraster(data = hillmulti,
              aes(fill = hillshade),
              show.legend = FALSE),
  scale_fill_distiller(palette = "Greys", na.value = NA),
  new_scale_fill(),
  geom_spatraster(data = mdt,
              aes(fill = alt),
              alpha = .7),
  scale_fill_hypso_tint_c(breaks = c(180, 250, 500, 1000,
                                     1500,  2000, 2500,
                                     3000, 3500, 4000))
  ) |> blend("multiply") +
  geom_sf(data = lagos_grandes,
          fill = "#c6dbef", colour = NA) +
  guides(fill = guide_colorsteps()) +
  labs(fill = "m") +
  coord_sf() +
  theme_void() +
```

```
      theme_void() +
        theme(legend.position = "bottom",
        legend.key.height = unit(.5, "lines"),
        legend.key.width = unit(4, "lines"),
        legend.title.position = "right")

plot(m)
```

Camas hospitalarias en Europa

La tarea consiste en crear un mapa de la UE con datos sobre camas hospitalarias a nivel regional (NUTS-2) (dataset hlth_rs_bdsrg) usando el paquete `{eurostat}`. La dos funciones fundamentales son `get_eurostat()` y `get_eurostat_geospatial()`.

```{r}
# Instalar los paquetes si no están instalados
if (!requireNamespace("eurostat", quietly = TRUE))
install.packages("eurostat")
if (!requireNamespace("ggplot2", quietly = TRUE))
install.packages("ggplot2")
if (!requireNamespace("sf", quietly = TRUE)) install.packages("sf")
if (!requireNamespace("dplyr", quietly = TRUE)) install.packages("dplyr")

# Cargar los paquetes
library(eurostat)
library(ggplot2)
library(sf)
library(dplyr)

```

Una vez instaladas las librerías que necesitamos vamos a obtener los datos de las camas hospitalarias a nivel regional (NUTS-2) empleando la función get_eurostat()

```{r}
# Para ello vamos a emplear la función get_eurostat
hospital_beds <- get_eurostat(id = "hlth_rs_bdsrg", time_format = "num")

```

Estudiamos como son los valores de las columnas

```{r}
str(hospital_beds)
head(hospital_beds)
```
Vamos a seleccionar el último año disponible de camas para que sea lo más actual posible y limpiamos los datos:

```{r}
hospital_beds_filtered <- hospital_beds %>%
  filter(!is.na(values), TIME_PERIOD == max(TIME_PERIOD)) %>%
```

```
  filter(unit == "P_HTHAB") # Comando útil para indicar las camas por cada
100.000 habitantes
```

Analizamos los datos que tenemos de los datos filtrados:

```{r}
str(hospital_beds_filtered)
unique(hospital_beds_filtered$geo)

```
Obtenemos los datos geoespaciales de las regiones NUT-2 mediante la
función get_eurostat_geospatial
```{r}
library(giscoR) #he tenido que instalar lalibrería para que me cargue todo
bien
nuts2_map <- get_eurostat_geospatial(nuts_level = 2, resolution = "20",
year = "2021")

#Unimos estos datos geoespaciales con las camas de los hospitales

map_data <- nuts2_map %>%
  left_join(hospital_beds_filtered, by = c("NUTS_ID" = "geo"))

```

Una vez que tenemos los datos que estábamos buscando creamos la
visualización con la ya conocida librería ggplot2:


```{r}
library(ggplot2)
library(sf)

# Visualización optimizada
camas_2021 <- ggplot(data = map_data) +
  geom_sf(aes(fill = values), color = "white", size = 0.1) +
  scale_fill_viridis_c(
    option = "plasma",
    na.value = "grey90",
    name = "Camas por\n100,000 habitantes"
  ) +
  theme_minimal(base_size = 8) +  # He puesto un tamaño de la base pequeño
para poder apreciar bien los países de los que tenemos datos
  labs(
    title = "Camas hospitalarias por región en la UE (NUTS-2)",
    subtitle = paste("Último año disponible:",
max(hospital_beds_filtered$TIME_PERIOD)),
    caption = "Fuente: Eurostat (hlth_rs_bdsrg)"
  ) +
  theme(
    plot.title = element_text(size = 20, face = "bold", hjust = 0.5),  #
Centrado y grande
    plot.subtitle = element_text(size = 16, hjust = 0.5),
    legend.title = element_text(size = 14),
    legend.text = element_text(size = 12),
    legend.position = "bottom",
    legend.key.width = unit(2.5, "cm"),
```

```
    legend.key.height = unit(0.5, "cm")
  ) +
  coord_sf(
    xlim = c(-40, 60),

    ylim = c(30, 75),
    expand = FALSE
  ) +
  guides(fill = guide_colorbar(barwidth = 20, barheight = 0.8))
plot(camas_2021)
```
```{r}
ggsave("camas_2021.png", camas_2021,
       height = 6, width = 12,
       units = "in",
       bg = "white")
```