CampoGausiano_animales.R

DairXP

2025-09-17

```
# Cargar
library(haven)
library(dplyr)
```

```
##
## Adjuntando el paquete: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(sf)
```

```
## Linking to GEOS 3.13.0, GDAL 3.10.1, PROJ 9.5.1; sf_use_s2() is TRUE
```

```
library(sp)
library(raster)
```

```
##
## Adjuntando el paquete: 'raster'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
library(gstat)
library(leaflet)
library(RColorBrewer)
library(terra)   # usamos terra para reproyección segura de raster
```

```
## terra 1.8.60
```

```r
# 1) Leer datos ------------------------------------------------------------
caratula <- read_sav("CARATULA.sav")      # contiene LATITUD/LONGITUD
cap400a_09 <- read_sav("09_CAP400A.sav")  # contiene ANIO, CCDD, CCPP, CCDI, P401, etc

# 2) Filtrar PUNO ----------------------------------------------------------
caratula_puno <- caratula %>% filter(NOMBREDD == "PUNO")
cap400a_puno <- cap400a_09 %>% filter(NOMBREDD == "PUNO")

# 3) Unir coordenadas ------------------------------------------------------
puno_merged <- cap400a_puno %>%
  left_join(dplyr::select(caratula_puno,
                          ANIO, CCDD, CCPP, CCDI, NSEGM, ID_PROD, UA,
                          LATITUD, LONGITUD),
            by = c("ANIO", "CCDD", "CCPP", "CCDI", "NSEGM", "ID_PROD", "UA"))

# 4) Limpiar coordenadas y convertir a sf ----------------------------------
puno_merged <- puno_merged %>% filter(!is.na(LATITUD) & !is.na(LONGITUD))
puno_sf <- st_as_sf(puno_merged, coords = c("LONGITUD","LATITUD"), crs = 4326, remove = FALSE)

# 5) Filtrar VACUNOS (P401 == "1") y crear valor = 1 por predio ------------
puno_sf$P401 <- as.character(puno_sf$P401)
puno_vacunos <- puno_sf %>% filter(P401 == "1")
if(nrow(puno_vacunos) == 0) stop("No se encontraron registros con P401 == '1' en PUNO.")
puno_vacunos <- puno_vacunos %>% mutate(valor = 1)

# 6) Transformar a UTM 32719 para métricas en metros -----------------------
puno_vac_utm <- st_transform(puno_vacunos, crs = 32719)
vacunos_sp <- as(puno_vac_utm, "Spatial")   # SpatialPointsDataFrame con campo 'valor'

# 7) Crear raster de conteo por celda (raw counts) -------------------------
cellsize <- 2000   # 2 km
buffer_m <- 5000   # 5 km buffer alrededor de puntos
bb <- st_bbox(puno_vac_utm)
xmin <- bb["xmin"] - buffer_m
xmax <- bb["xmax"] + buffer_m
ymin <- bb["ymin"] - buffer_m
ymax <- bb["ymax"] + buffer_m

nx <- ceiling((xmax - xmin) / cellsize)
ny <- ceiling((ymax - ymin) / cellsize)
r_template <- raster(xmn = xmin, xmx = xmin + nx*cellsize,
                     ymn = ymin, ymx = ymin + ny*cellsize,
                     crs = CRS("+init=epsg:32719"),
                     resolution = cellsize)
```

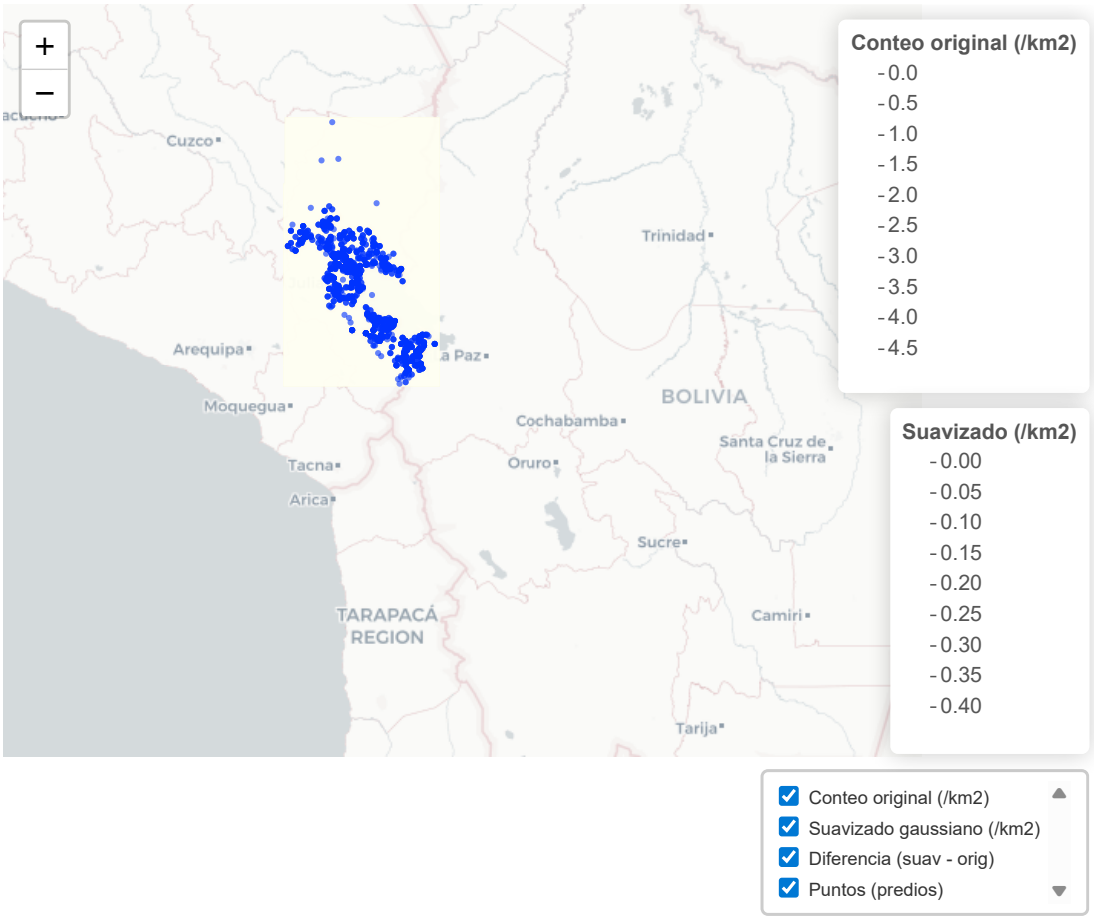
```
## Warning in CPL_crs_from_input(x): GDAL Message 1: +init=epsg:XXXX syntax is
## deprecated. It might return a CRS with a non-EPSG compliant axis order.
```

```r
# Rasterizar: contar puntos por celda (suma de 'valor', que es 1 por predio)
vac_count_r <- rasterize(vacunos_sp, r_template, field = "valor", fun = sum, background = 0)

# 8) Suavizado gaussiano (filtro focal) ------------------------------------
# sigma en metros (desviación): ajustar según cuánto quieres suavizar. Ej: 3000 m
```

```r
sigma <- 3000
w <- focalWeight(vac_count_r, d = sigma, type = "Gauss")
vac_density_r <- focal(vac_count_r, w = w, pad = TRUE, padValue = 0, na.rm = TRUE)

# 9) Normalizar opcional: densidad por km2 ----------------------------------
area_km2 <- (cellsize * cellsize) / 1e6
vac_density_per_km2 <- vac_density_r / area_km2
vac_count_per_km2 <- vac_count_r / area_km2   # también convertimos conteo original a /km2 para compara

# 10) Diferencia (suavizado - original) -------------------------------------
diff_r <- vac_density_per_km2 - vac_count_per_km2

# 11) Reproyectar a WGS84 para Leaflet usando 'terra' (evitamos rgdal) ------
# Convertir a SpatRaster (terra)
spat_raw <- terra::rast(vac_count_per_km2)
spat_smooth <- terra::rast(vac_density_per_km2)
spat_diff <- terra::rast(diff_r)

# Reproyectar a EPSG:4326 con método bilinear para continuos
spat_raw_wgs <- terra::project(spat_raw, "EPSG:4326", method = "near")        # counts -> nearest
spat_smooth_wgs <- terra::project(spat_smooth, "EPSG:4326", method = "bilinear")
spat_diff_wgs <- terra::project(spat_diff, "EPSG:4326", method = "bilinear")

# Convertir de nuevo a RasterLayer (para compatibilidad con addRasterImage)
raw_wgs_r <- raster::raster(spat_raw_wgs)
smooth_wgs_r <- raster::raster(spat_smooth_wgs)
diff_wgs_r <- raster::raster(spat_diff_wgs)

# 12) Paletas y valores (filtrar NA)
vals_raw <- values(raw_wgs_r); vals_raw <- vals_raw[!is.na(vals_raw)]
vals_smooth <- values(smooth_wgs_r); vals_smooth <- vals_smooth[!is.na(vals_smooth)]
vals_diff <- values(diff_wgs_r); vals_diff <- vals_diff[!is.na(vals_diff)]

pal_raw <- colorNumeric("YlGnBu", domain = vals_raw, na.color = "transparent")
pal_smooth <- colorNumeric("YlOrRd", domain = vals_smooth, na.color = "transparent")
# Para diferencia usamos paleta centrada en cero
max_abs_diff <- max(abs(vals_diff), na.rm = TRUE)
pal_diff <- colorNumeric(colorRampPalette(c("blue","white","red"))(11),
                         domain = c(-max_abs_diff, max_abs_diff), na.color = "transparent")

# 13) Puntos en lon/lat para mostrar en mapa
points_wgs <- st_transform(puno_vac_utm, 4326)

# 14) Mapa leaflet con control de capas -------------------------------------
m <- leaflet() %>%
  addProviderTiles("CartoDB.Positron") %>%
  addRasterImage(raw_wgs_r, colors = pal_raw, opacity = 0.6, group = "Conteo original (/km2)") %>%
  addRasterImage(smooth_wgs_r, colors = pal_smooth, opacity = 0.6, group = "Suavizado gaussiano (/km2)"
  addRasterImage(diff_wgs_r, colors = pal_diff, opacity = 0.7, group = "Diferencia (suav - orig)") %>%
  addCircleMarkers(data = points_wgs, radius = 2, stroke = FALSE, fillOpacity = 0.6, group = "Puntos (p
  addLegend(pal = pal_raw, values = vals_raw, title = "Conteo original (/km2)", group = "Conteo original
  addLegend(pal = pal_smooth, values = vals_smooth, title = "Suavizado (/km2)", group = "Suavizado gauss
  addLayersControl(
```

```
    overlayGroups = c("Conteo original (/km2)", "Suavizado gaussiano (/km2)", "Diferencia (suav - orig)
    options = layersControlOptions(collapsed = FALSE)
  )

# Mostrar mapa
m
```

```
## PhantomJS not found. You can install it with webshot::install_phantomjs(). If it is installed, pleas
```




```
# (Opcional) Guardar rasters en disco
# writeRaster(raw_wgs_r, "vac_count_per_km2_wgs84.tif", overwrite = TRUE)
```

```
# writeRaster(smooth_wgs_r, "vac_density_per_km2_wgs84.tif", overwrite = TRUE)
# writeRaster(diff_wgs_r, "vac_density_diff_wgs84.tif", overwrite = TRUE)
```