DS221

Data Visualization and Storytelling

PORTFOLIO 1 - Final Report

ITALY:

A Population Density Map

APUYA, Marlou P.

PABOLOLOT, Honey Angel H.

SANCHEZ,Airyll H.

VALIENTE,April Rose D.

April 08, 2024

How to Create a 3D Population Density Map in R

1. Install Required Packages:

You need to install the necessary packages. Run the following commands in your R console. Try to install them one by one, it might require restarting the R-session several times.

```
install.packages("sf", dependencies=TRUE)
install.packages("tmap", dependencies=TRUE)
install.packages("mapview", dependencies=TRUE)
install.packages("stars", dependencies=TRUE)
install.packages("rayshader", dependencies=TRUE)
install.packages("MetBrewer", dependencies=TRUE)
install.packages("rayrender")
install.packages("extrafont", dependencies=TRUE)
install.packages("magick", dependencies=TRUE)
```

In this chunk, we installed(run) the following packages as it is, and we haven't encountered any problems so far.

2. Load Packages and Set Options:

Load the required libraries and set the RGL options:

```
options(rgl.useNULL = FALSE)
require(tidyverse)
require(sf)
require(tmap)
require(ggplot2)
require(mapview)
require(stars)
require(rayshader)
require(MetBrewer)
require(colorspace)
require(rayrender)
require(magick)
require(extrafont)
```

There's no revision done in this chunk. All the functions run well with no errors encountered.

3. Load and Transform Data:

You'll need to load the population data and administrative boundaries for the Philippines, transforming them into a suitable coordinate system. The data is downloaded from Kontur Population.

```
ph_hex <- st_read("data/kontur_population_PH_20231101.gpkg") %>%
st_transform(3106)
```

```
ph_admin <- st_read("data/kontur_boundaries_PH_20230628.gpkg") %>%
st_transform(3106)
```

In this chunk we just changed our dataset which was downloaded from Kontur Population, so Ph was replaced with Italy thus "ph_hex" to "italy_hex". We place our data in the same folder as our RMD file so that we can directly access the dataset without needing to add a folder name.

4. Check and Create Boundaries:

Inspect the 'name_en' column and create the boundary for the Philippines. Use the filter option to plot specific districts and divisions on the map.

```
distinct_names <- ph_admin %>% distinct(name_en)
print(distinct_names)

# Creating BD Boundary
ph_boundary <- ph_admin %>%
  st_geometry %>%
  st_union %>%
  st_sf %>%
  st_make_valid()
```

In this chunk, we modified the country from Ph to Italy, and there were no other errors encountered. The 'name_en' is a list of cities in our chosen country. You can create a boundary by filtering either your chosen city or the entire country using the filter function

5.Plot Boundaries for Verification:

Visualize the hex data and boundaries to ensure accuracy.

```
names(ph_hex)

ggplot(ph_hex) +
  geom_sf(aes(fill = population), color = "gray66", linewidth = 0) +
  geom_sf(data = ph_boundary, fill = NA, color = "black")
```

In this chunk, the unnecessary function was removed ("ph_hex") and changes were followed, which was replacing the parameter "fill" with "color" This is because the original function for color doesn't show the intended representation of the population.

6. Calculate Aspect Ratio:

Determine the aspect ratio for the map based on the bounding box of the boundary.

```
# setting the ph boundary as a bounding box
bbox <- st_bbox(ph_boundary)

# finding the aspect ratio
bottom_left <- st_point(c(bbox[["xmin"]], bbox[["ymin"]])) %>%
  st_sfc(crs = 3106)
```

```
bottom_right <- st_point(c(bbox[["xmax"]], bbox[["ymin"]])) %>%
  st_sfc(crs = 3106)
top_left <- st_point(c(bbox[["xmin"]], bbox[["ymax"]])) %>%
  st_sfc(crs = 3106)
top_right <- st_point(c(bbox[["xmin"]], bbox[["ymax"]])) %>%
  st_sfc(crs = 3106)

width <- st_distance(bottom_left, bottom_right)
height <- st_distance(bottom_left, top_left)

if(width > height) {
  w_ratio = 1
  h_ratio = height / width
} else {
  h_ratio = 1.1
  w_ratio = width / height
}
```

In this chunk, we modified the country from Ph to Italy, and there were no other errors encountered.

7. Rasterize Population Data:

Convert the population data into a raster format suitable for 3D rendering.

- For interactively checking the 3D plot setting the size low will help render in real time.
- To improve the quality of the 3D image when saving, change the settings to a higher resolution.

```
# convert to raster to convert to matrix
size = 3500

pop_raster <- st_rasterize(
  ph_hex,
  nx = floor(size * w_ratio) %>% as.numeric(),
  ny = floor(size * h_ratio) %>% as.numeric()
)

pop_matrix <- matrix(pop_raster$population,
                     nrow = floor(size * w_ratio),
                     ncol = floor(size * h_ratio))
```

In this chunk, we modify the country from Ph to Italy, and this was the only error encountered.

8. Define Color Palette:

Select a color palette from the MetBrewer or RColorBrewer library and customize it for your map.

```
# Create color palette from MetBrewer Library
color <- MetBrewer::met.brewer(name="OKeeffe1", direction = -1)

tx <- grDevices::colorRampPalette(color, bias = 4.5)(256)
swatchplot(tx)
swatchplot(color)
```

.

In this chunk, the direction of the color palette is changed into a positive one, instead of a negative one, and this is because the negative one gives off a dim impression, compared to the positive one, thus the latter was chosen so that the visual can be presented more accurately. We also change the bias to properly display the variety of colors of our chosen palette.

9. Render 3D Map:

Use Rayshader to create a 3D representation of the population density.

```
# Close any existing 3D plot before plotting another
rgl::close3d()

pop_matrix %>%
  height_shade(texture = tx) %>%
  plot_3d(heightmap = pop_matrix,
          #zscale = 250 / 4.5,
          zscale = 25,
          solid = F,
          shadowdepth = 0.8
          )

# Adjusting Camera Angle
render_camera(theta = 0,
              phi = 70,
              zoom = 0.55,
              fov = 100
)


# To interactively view the 3D plot
rgl::rglwidget()
```

The parameters used in # Adjusting Camera Angle (theta, phi, zoom and fov) are modified. This is because, whenever the 3D Map is displayed it only shows the front angle and does not satisfy the accurate representation of the map.

10. Render in high-quality and Save Image:

Fine-tune the camera angle and render a high-quality image of the 3D map.

```
outfile <- glue::glue("Plots/Ph_final_2.png")

{
  start_time <- Sys.time()
  cat(crayon::cyan(start_time), "\n")
  if(!file.exists(outfile)) {
    png::writePNG(matrix(1), target = outfile)
  }

  render_highquality(
    filename = outfile,
    #interactive = F,
    lightdirection = 50, #Degree
    #lightaltitude = c(30, 80),
    #lightcolor = c(subset_colors[4], "white"),
    lightcolor = c("white", "white"),  # Set both lights to white
    lightintensity = c(600, 100),
    width = 1980,
    height = 1180,
    #width = 1000,
    #height = 1400,
    samples = 550
    #samples = 2
  )

  end_time <- Sys.time()
  diff <- end_time - start_time
  cat(crayon::cyan(diff), "\n")
}
```

In this chunk, major revision was done. This is due to the fact that the original parameters don't present the colors accurately, and thus HDR was used. Plots are also made to automatically save the rendered map into its designated folder and parallel is also added for faster rendering.

11. Annotate the image

You can add names and more details about your generated visualization.

```r
# --------------------------Anotate
# Install and load the showtext package
install.packages("showtext")
library(showtext)
install.packages("extrafont")
library(extrafont)
font_import(pattern = "Philosopher")

pop_raster <- image_read("Plots/Ph_final.png")

text_color <- darken(color[1], .4)
swatchplot(text_color)


# Automatically enable font support
showtext_auto()

# Download and register the Philosopher font from Google Fonts
font_add_google("Philosopher", regular = "400", bold = "700")

pop_raster %>%
  image_annotate("Philippines",
                 gravity = "northeast",
                 location = "+50+50",
                 color = text_color,
                 size = 120,
                 font = "Philosopher",
                 weight = 700,
                 # degrees = 0,


  ) %>%
  image_annotate("POPULATION DENSITY MAP",
                 gravity = "northeast",
                 location = "+50+175",
                 color = text_color,
                 size = 28.5,
                 font = "Philosopher",  # Corrected font name
                 weight = 500,
                 # degrees = 0,
  ) %>%
  image_annotate("Visualization by: Insert Name \nData: Kontur Population
2023",
                 gravity = "southwest",
                 location = "+20+20",
                 color = alpha(text_color, .8),
                 font = "Philosopher",  # Corrected font name
                 size = 25,
                 # degrees = 0,
```

```
  ) %>%
  image_write("Plots/Annotated_plot_ph.png", format = "png", quality = 100)
```

A few revisions are done in this chunk, firstly the parameters “gravity”, “location”, and “size” are modified. We also added labels indicating the most populous cities on our rendered map . Lastly, the parameter “quality” is modified to 1000 instead of 100.