Historical Persistence

Applied Economics Research Course

Bas Machielsen
Utrecht University
2023-12-14

Some Common Data Sources

Data Sources

- Global Agro-Ecological Zones
- Gridded population of the world
- Gridded population of the Netherlands
- Road Network Netherlands
- Worldwide Biodiversity Data
- Urban Land Cover Data (Europe only)
- Urban Land Cover Data
- Biodiversity datasets (Europe only)

- Some datasets might require registration at the proprietor

Some Common Data Wrangling Operations

Import `.h5` nightlights data

- Some nightlights data is structured as `.h5` data
- These are "layered" raster files that contain a potentially large amount of variables
- I will demonstrate how this works
- Furthermore, I will also show how to *combine* different raster files

Import `.h5` data

- A `.h5` dataset can be imported in the following way:
  - Load the `rhdf5` library:

```r
if (!require("BiocManager", quietly = TRUE))
    install.packages("BiocManager")

BiocManager::install("rhdf5")
```

- Extract the metadata, and read where the latitude, longitude data are contained

```r
library(rhdf5)
meta <- rhdf5::h5ls('VNP46A4.A2022001.h18v03.001.2023082112129.h5')
lat <- rhdf5::h5read('VNP46A4.A2022001.h18v03.001.2023082112129.h5',
                     name='/HDFEOS/GRIDS/VIIRS_Grid_DNB_2d/Data Fields/lat')
lon <- rhdf5::h5read('VNP46A4.A2022001.h18v03.001.2023082112129.h5',
                     name='/HDFEOS/GRIDS/VIIRS_Grid_DNB_2d/Data Fields/lon')
```

Import `.h5` data

- Then, select one layer of the map you are interested in
  - I also import a shapefile of the Netherlands because I want to pay attention to this part
  - Set the `extent` of a raster to the latitude and longitude data
  - Set the `crs` of a raster to WGS84 (the default projection of the NASA VIIRS data)

```
netherlands <- geodata::gadm("Netherlands", path="./") |> st_as_sf()
raster <- terra::rast('VNP46A4.A2022001.h18v03.001.2023082112129.h5') |>
  terra::subset("AllAngle_Composite_Snow_Covered")

ext(raster) <- c(min(lon), max(lon), min(lat), max(lat))
crs(raster) <- crs('wgs84')
```

Import the `.h5` data

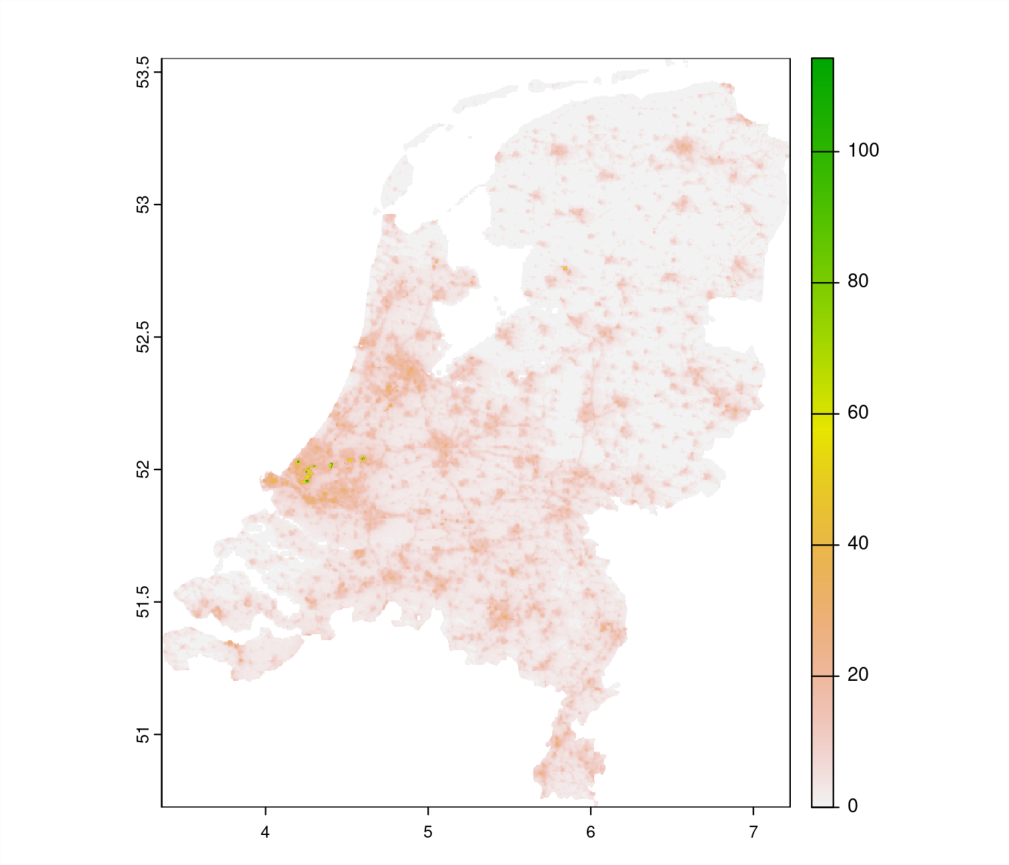
- Now, the data is usable in principle
- However, we do some post-processing:
  - We first mask (confine) the data set to the extent (overlap) with the Netherlands
  - Then we crop it so as to remove unnecessary `NA` data
  - Finally, we filter out the default sea level

```
masked_raster <- terra::mask(raster, netherlands)
netherlands_nightlights <- crop(masked_raster, netherlands)
netherlands_nightlights[netherlands_nightlights == 65535] = NA
```

Plot the outcome

- Finally, we can plot the outcome

```
terra::plot(sqrt(netherlands_nightlights))
```



Combine various raster datasets

- By writing a function, you can also combine several datasets:

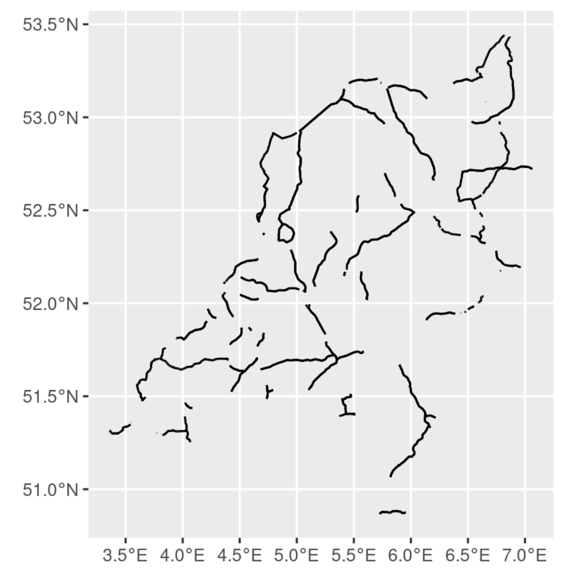
```r
combine_and_clean_rasters <- function(directory_with_rasters,
                                      type="AllAngle_Composite_Snow_Covered"){
  # Find all the datasets in the directory
  nms <- list.files(directory_with_rasters, pattern='\\.h5')
  files <- paste0(directory_with_rasters, nms)
  # Pick one type of map
  rasters <- map(files, ~ terra::rast(.x) |>
                   terra::subset(type))
  # Extract lat, lon
  lat <- map(files, ~ rhdf5::h5read(.x,
                   name='/HDFEOS/GRIDS/VIIRS_Grid_DNB_2d/Data Fields/lat'))
  lon <- map(files, ~ rhdf5::h5read(.x,
                   name='/HDFEOS/GRIDS/VIIRS_Grid_DNB_2d/Data Fields/lon'))
  rasters <- imap(rasters, ~ {
  ext(.x) <- c(min(lon[[.y]]), max(lon[[.y]]), min(lat[[.y]]), max(lat[[.y]]))
  .x  })
  # Convert to correct CRS
  rasters <- map(rasters, ~ { crs(.x) <- crs('wgs84'); .x })
  out <- do.call(terra::mosaic, rasters)
  return(out)
}
```

Road Density

- Now, I show how to combine the `roads` data to construct density

```
netherlands <- geodata::gadm("Netherlands", level=2, path="./") |> st_as_sf()
roads <- st_read('./nwb_hoofdwegen.gpkg') |> st_transform(crs=crs(netherlands)) |>
```

```
roads |> ggplot() + geom_sf()
```

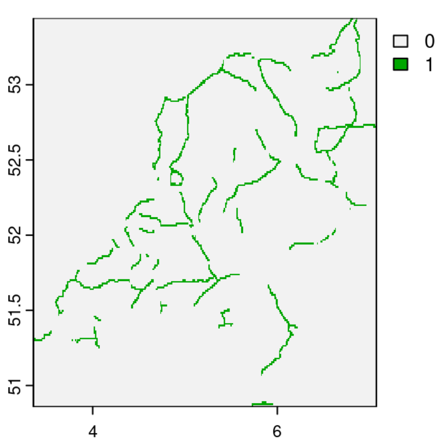


Buffer the roads

- I build a buffer of 10 meters around each road
    - Rasterize it
    - Then set all values where there is a road to 1 (all roads are equal)
- Finally, plot it:

```
roads_buffer <- st_buffer(roads, dist=10)
roads_raster <- st_rasterize(roads_buffer,
                             options = "ALL_TOUCHED=TRUE")

roads_raster[roads_raster > 0] = 1; roads_raster[is.na(roads_raster)] = 0
final_raster <- terra::rast(roads_raster); terra::plot(final_raster)
```



Aggregate the roads to a shapefile

- We already know how to do this - we can use the `extract` function from `terra`

```r
values <- terra::extract(final_raster, netherlands) |>
  janitor::clean_names()

road_density <- values |>
  group_by(id) |>
  summarize(mean_roads = mean(id_2, na.rm=T))
```

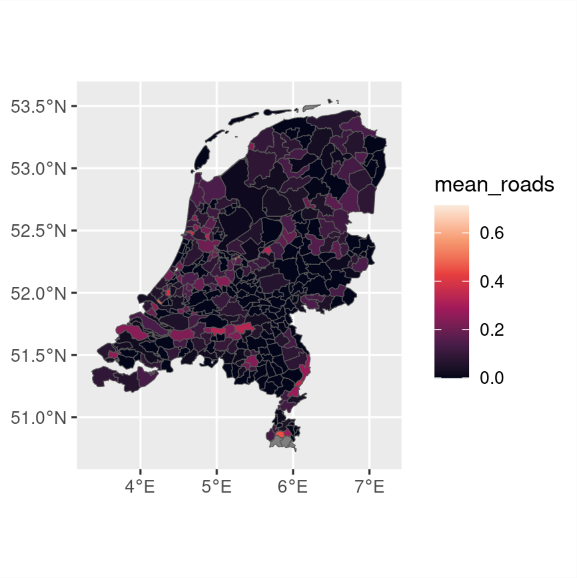
- Finally, we merge the road density to the shapefile:

```r
nl_roads <- netherlands |>
  mutate(ID=row_number()) |>
  left_join(road_density, by=c("ID" = "id"))
```

Plot the output

- We end up with a road density per municipality:

```
nl_roads |>
  ggplot(aes(fill=mean_roads)) +
  geom_sf() +
  scale_fill_viridis_c(option='F')
```



Land Cover Per Municipality

Land Cover

- Download a map from e.g. here
    - I downloaded the CORINE Land Cover 2018 Here
- Import it (it is hidden in a .zip within a .zip)
- And put everything in the same projection:

```
netherlands <- geodata::gadm("Netherlands", level=2, path="./") |> st_as_sf()
land_cover <- terra::rast('./u2018_clc2018_v2020_20u1_raster100m/DATA/U2018_CLC201
netherlands <- st_transform(netherlands, st_crs(land_cover))
```

Aggregate categories per polygon

- Implement the following `for`-loop:

```r
# Create an empty data frame to store results
results <- data.frame()

# Loop through each polygon
for (i in 1:nrow(netherlands)) {
  polygon <- netherlands[i,]
  # Clip the raster to the current polygon
  raster_clipped <- terra::crop(land_cover, polygon)
  # Calculate the area covered by each value within the polygon
  extracted_values <- extract(raster_clipped, polygon)
  out <- extracted_values |>
    group_by(LABEL3) |> count() |> ungroup() |>
    mutate(tot = sum(n))   |>
    group_by(LABEL3) |>
    mutate(percentage = n/tot) |> dplyr::select(-c(n, tot)) |>
    pivot_wider(names_from=LABEL3, values_from = percentage)

  # Add polygon ID and coverage percentages to results
  results <- bind_rows(results, out)
}
```

Output

- What you end up with is this:

```
results ▷
  select(c(1:4)) ▷
  head(5)
```

```
##   Discontinuous urban fabric Industrial or commercial units Mineral extraction sites
## 1                 0.03603057                    0.002356806             0.0008213112
## 2                 0.21150855                    0.041272880                       NA
## 3                 0.05453946                    0.007074002             0.0009163215
## 4                 0.03574292                    0.014564404                       NA
## 5                 0.02402786                    0.003218411             0.0022925668
```

Merge

- Now, we can put this together in Netherlands by merging on the basis of row number..:
    - Beforehand, we fill in `NA` observations with zero:

```
results ← results ▷ mutate(across(everything(), ~ replace_na(.x, 0)))
netherlands ← netherlands ▷ bind_cols(results)
```

- Let's plot the most urban municipalities:

```
netherlands ▷
  ggplot(aes(fill=`Discontinuous urban fabric`)) + geom_sf()
```