world_boundaries

Marco Giovanni Ferrari Prasiddha Rajaure Stefano Ciampa

2025-03-24

Introduction - A trade-off between proximity and geopolitics

In our network, nodes are single countries or territorial area. However, many countries are composed by territories that may be distant from one another, such as France and its oversees territories. This raises a trade-off between physical geography and geopolitics. In fact, considering very distant territories as a single geographical entity may produce an unrealistic map of the cable-data network. At the same time, considering as separate entities countries that are by nature composed by archipelagos would produce a poorly readable map, which would also not reflect the real political borders. For this reason, we break the trade-off by aggregating territories that, conditional on being part of the same authority (e.g., the France Republic), are within a distance of less than 100 Km. This distance is set at the beginning of the code:

```
distance_threshold <- 100000 # 100 km
```

In addition, we will require the following libraries for this analysis:

```
library(sf)
library(dplyr)
library(tidyr)
library(kableExtra)
library(ggplot2)
library(igraph)
```

We start by importing the dataset[1] and showing how the original data are structured:

```
world_boundaries_path <- "raw_data/geopolitical_data/world-administrative-boundaries/"

world_boundaries <- st_read(paste0(world_boundaries_path,
                                   "world-administrative-boundaries.shp"),
                            quiet = T) %>%
  select(iso3, status,color_code, region, iso_3166_1_)

world_boundaries %>%
  head(8) %>%
  kable(format = "latex", digits = 2, caption = "World Boundaries Dataset") %>%
  kable_styling(full_width = TRUE,
                bootstrap_options = c("striped", "hover", "condensed", "responsive"),
                latex_options = "hold_position")
```

[1] Source: https://public.opendatasoft.com/explore/dataset/world-administrative-boundaries/table/

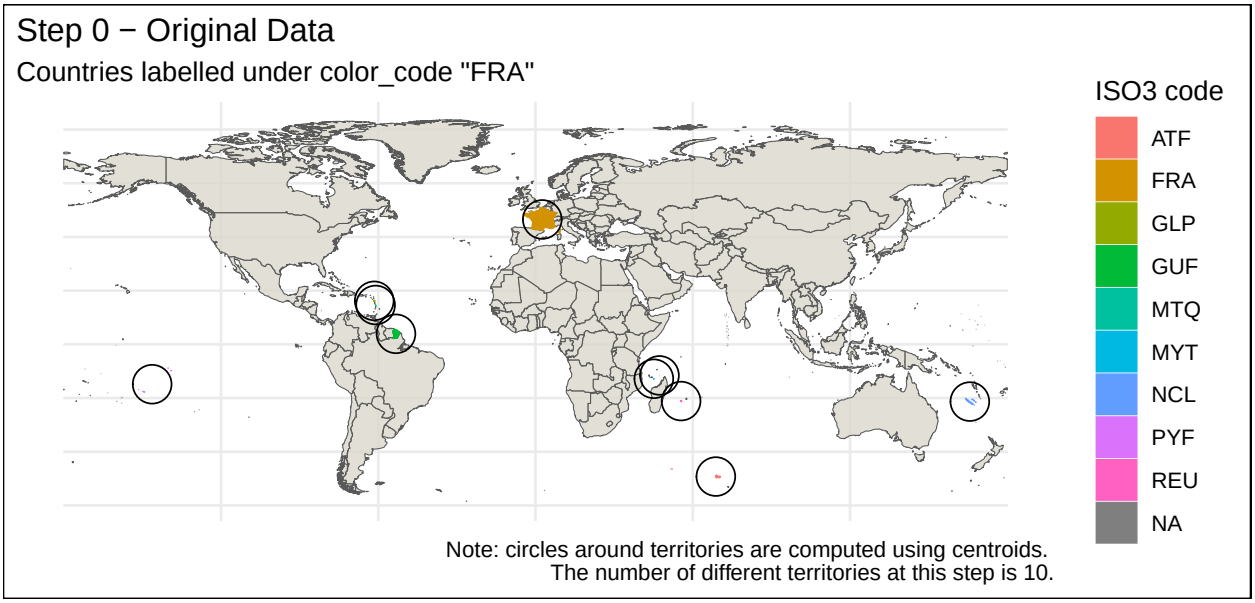
Table 1: World Boundaries Dataset

iso3	status	color_code	region	iso_3166_1_	geometry
MNP	US Territory	USA	Micronesia	MP	MULTIPOLYGON (((145.6333 14...
NA	Sovereignty unsettled	RUS	Eastern Asia	NA	MULTIPOLYGON (((146.6827 43...
FRA	Member State	FRA	Western Europe	FR	MULTIPOLYGON (((9.4475 42.6...
SRB	Member State	SRB	Southern Europe	RS	MULTIPOLYGON (((20.26102 46...
URY	Member State	URY	South America	UY	MULTIPOLYGON (((-53.3743 -3...
GUM	US Non-Self-Governing Territory	GUM	Micronesia	GU	MULTIPOLYGON (((144.7094 13...
PAN	Member State	PAN	Central America	PA	MULTIPOLYGON (((-81.67847 7...
ANT	NL Territory	NLD	Caribbean	NA	MULTIPOLYGON (((-68.19736 1...

Aggregating Territories

Step 0 - Original Data

The initial dataset provides the iso3 code for geographical entities, which should be sufficient to uniquely identify territories. However, being based on political principles, it may be unsuited to distinguish between territories that are distant to one another. The first Figure considers the case of France. As an example, Corsica is cast in the original data as part of the France mainland territories (same `iso3`), without any particular consideration on the effective geographical proximity. We will see in the last step that these are separated by more than 100Km of seas and for this reason should be considered as separate nodes for cable landing points. To address the need of having an allocation of territories that depends on proximity but preserves the geopolitical adherence of a given territory, we leverage on the variable `color_code`, which defines, for each territory, the authority (country) that effectively controls that region.



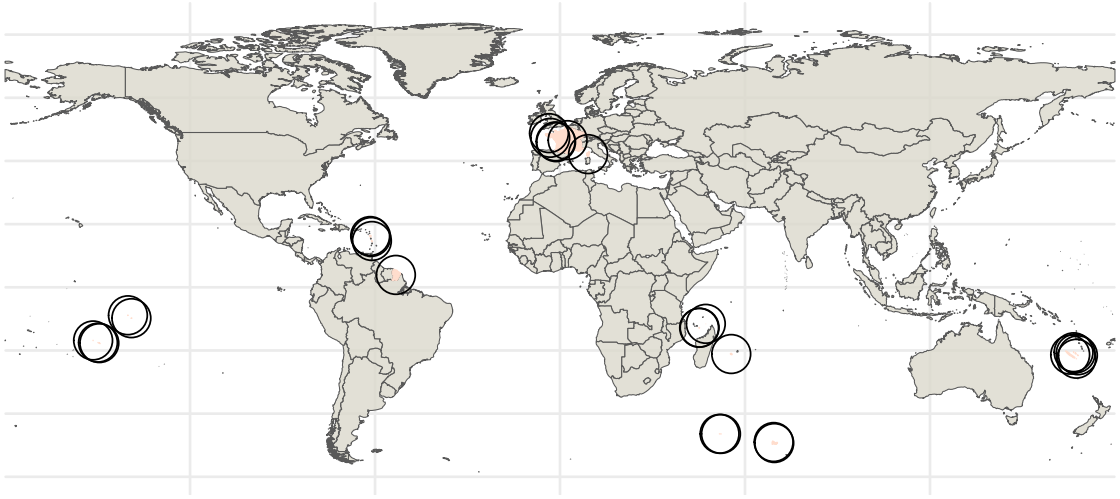
Step 1 - Separation in Single Territories

In this step, we separate different territories independently of their distance. In this way we obtain a large dataset collecting all the (unconnected) territories of each authority.

```
world_boundaries <- world_boundaries %>%
  select(color_code, geometry) %>%
  rename(country_code = color_code)

world_boundaries <- world_boundaries %>% group_by(country_code)  %>%
  summarise(geometry = st_union(geometry))

world_boundaries <- world_boundaries %>% st_cast("MULTIPOLYGON")

world_boundaries_single_territories <- world_boundaries %>%
  st_cast(to = "POLYGON")
```

Step 1 − Separation in Single Territories

Countries labelled under color_code "FRA"



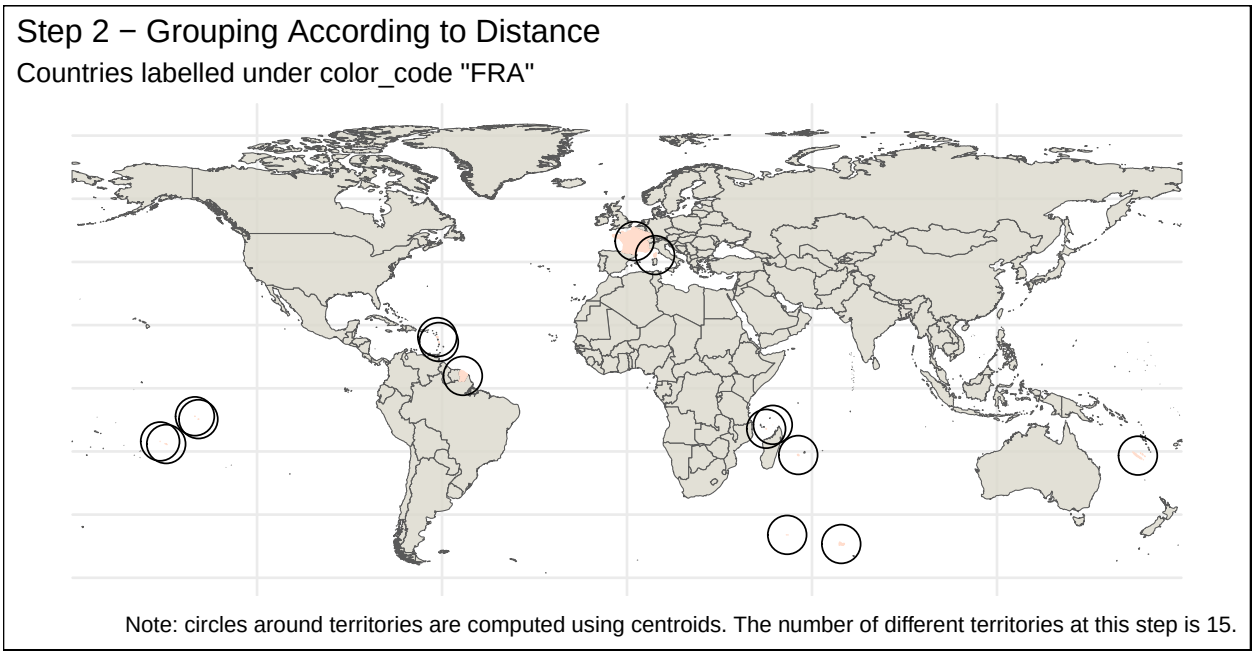
Note: circles around territories are computed using centroids. We omit the iso3 code as at this point has lost its meaning.
The number of different territories at this step is 28.

Step 2 - Grouping According to Distance

In this last step, we proceed to re-group territories provided that they fall within the distance of 100Km. To do so, we compute, for each authority (e.g., the France Republic), whether couples of territories are within the distance threshold. This returns a square and symmetric matrix made of logical values that are true if the two territories are sufficiently close. In doing so we use the function `st_is_within_distance()` embedded in package `sf`. We transform this matrix into an undirected graph, that allows us to group territories.

```
# Step 2
territories_list <- list()

for(country in unique(world_boundaries_single_territories$country_code)){
  territories <- world_boundaries_single_territories %>%
    filter(country_code == country)

  # Compute the distance matrix
  distance_matrix <- st_is_within_distance(territories,
                                           dist = distance_threshold)

  # Convert the logical matrix to an adjacency matrix
  adj_matrix <- as.matrix(distance_matrix)
  country_graph <- graph_from_adjacency_matrix(adj_matrix, mode = "undirected")

  # Find connected components
  components <- components(country_graph)

  # Add the component membership to the data frame
  territories$component <- components$membership

  # Save results in a list
```

```
  territories_list[[country]] <- territories
}

# Converting to dataframe
world_boundaries_single_territories <- do.call(rbind, territories_list)

# Merging the territories
world_boundaries_single_territories <- world_boundaries_single_territories %>%
  group_by(country_code, component) %>%
  summarise(geometry = st_union(geometry)) %>%
  ungroup() %>%
  mutate(country_code_n = paste0(country_code, "_", component)) %>%
  relocate(country_code_n, .before = country_code) %>%
  select(- component)
```



Saving Results

Finally, we store the dataset of aggregated territories to reuse them in the other parts of the project.

```
world_boundaries_single_territories %>%
  head(8) %>%
  kable(format = "latex", digits = 2, caption = "World Territories Dataset") %>%
  kable_styling(full_width = TRUE,
                bootstrap_options = c("striped", "hover", "condensed", "responsive"),
                latex_options = "hold_position")
```

Table 2: World Territories Dataset

country_code_n	country_code	geometry
ABW_1	ABW	POLYGON ((-69.88556 12.4577...
AFG_1	AFG	POLYGON ((74.83943 37.31975...
AGO_1	AGO	MULTIPOLYGON (((23.95403 -1...
AIA_1	AIA	POLYGON ((-63.14001 18.1683...
ALB_1	ALB	POLYGON ((20.05611 42.56291...
AND_1	AND	POLYGON ((1.7241 42.52139, ...
ARE_1	ARE	MULTIPOLYGON (((56.23804 25...
ARG_1	ARG	MULTIPOLYGON (((-62.65723 -...

```
saveRDS(world_boundaries_single_territories,
        "work_data/world_data_infrastructure/world_boundaries_single_territories.rds")
```