necountries: a R package to select and map a subset of countries

necountries is a small package that performs three tasks:

- constructing a `sf` with a relevant subset of countries,
- joining this `sf` with a tibble,
- providing a simple `plot` method to get a default plot.

Is is loaded using:

```
library(necountries)
```

and use shape files provided by the Natural Earth website and use extensively **sf** (Pebesma 2018) and **ggplot2** (Wickham 2016) and some other **R** packages[1].

What is a country ?

This is a most difficult question that it seems.[2] There are 193 member states of the United Nations and 2 general assembly observer states (Holly See and State of Palestine). Some countries have dependencies which are often overseas territories, like for example French Polynesia and New Caledonia for France, which have a special status somewhere between a normal region and a sovereign state. Finally, it is convenient, at least as far as drawing maps is concerned, to cut some countries in different pieces, the main territory and some parts. Consider for example Spain.

Spain consists of a main, continental territory and two sets of Islands, the Balearic Islands and the Canaries Islands, which are Spanish provinces. It's not a problem to plot the Balearic Islands along with continental Spain as they are almost entirely contained in the bounding

[1] In particular **ggrepel** (Slowikowski 2021) for labels and **classInt** (Bivand 2023) to compute automatically class intervals.

[2] In Massicotte and South (2023), a package that enables to download within **R** maps from naturalheart, there is a vignette on this question.

box of Continental Spain. It is not the case for Canaries Islands which are situated near the coasts of Morocco. Therefore, it is convenient to consider two different geometries for Spain: continental Spain and the Balearic on the one hand and the Canaries Islands on the other hand. This division doesn't obey to any political rule but is performed only for plotting convenience. In the same spirit, the United States of America is splitted in three (Mainland USA, Alaska and Hawaii) and Italy is kept as one geometry as Sicily and Sardinia are close enough to mainland Italy.

We'll therefore consider three categories of entities:

- main territory of sovereign countries (most of the time the whole country),
- parts of a sovereign country,
- dependencies of a sovereign country.

countries is based on **Natural Earth**, using the most detailed scale, which is `1:10,000,000` (1cm = 100km). **Natural Earth** provides different shape files containing the administrative borders of countries, with a different number of entities. They are called **sovereignty**, **countries**, **units** and **subunits**. For example, in the **sovereignty** file, France is a unique line (and therefore a unique multipolygon), as in the **countries** file, there is one line for each dependency (New Caledonia and French Pacific for example) and one unique line for continental France and the 5 overseas departments. In the **units** file, each overseas departments has its own line and in the **subunits** file Corsica also has its own line.

In the **countries** package, we start from the **countries** file, which contains 258 entities, and we split some of them to obtain finally 295 entities, categorized as follow:

- **main** (199): roughly corresponds to the main parts of the sovereign countries, ie the 193 UN recognized countries, the 2 UN observer countries (Palestine and Vatican), and 4 not or not fully recognized countries (Kosovo, Somaliland, Northern Cyprus and Taiwan),
- **part**: 37 parts of sovereign countries (including Macao and Hong Kong),
- **dependency**: 48 dependencies of sovereign countries,

- **indeterminate** : 11 territories, Western Sahara, Brazilian Island, Cyprus No Mans Area, Siachen Glacier, Southern Patagonian Ice Field, Bir Tawil, Antarctica, Spratly Islands, Bajo Nuevo Bank (Petrel Is.), Serranilla Bank, Scarborough Reef

The raw information about countries is stored in a `sf` called `ne_countries`[3]. The details of the computation are presented in the last section of this document. Two geometries are present in this `sf`, `polygon` for the administrative boundaries and `point` for the point coordinate of the capital:

```
ne_countries %>% as_tibble %>% select(- polygon, - point) %>%
    print(n = 2, width = Inf)
#> # A tibble: 295 x 19
#>   iso2  iso3  type  country     sovereign   capital status en
#>   <chr> <chr> <fct> <chr>       <chr>       <chr>   <chr>  <chr>
#> 1 AF    AFG   main  Afghanistan Afghanistan Kabul   member Afghanistan
#> 2 <NA>  <NA>  part  Agalega     Mauritius   <NA>    <NA>   <NA>
#>   fr          de          es                 it          region subregion
#>   <chr>       <chr>       <chr>              <chr>       <chr>  <chr>
#> 1 Afghanistan Afghanistan "Afganist\\u00e1n" Afghanistan Asia   Southern Asia
#> 2 <NA>        <NA>         <NA>              <NA>        Africa Eastern Africa
#>   wbregion        pop   gdp economy                    income
#>   <chr>         <dbl> <int> <chr>                      <chr>
#> 1 South Asia 38041754 19291 7. Least developed region 5. Low income
#> 2 <NA>             NA    NA <NA>                       <NA>
#> # i 293 more rows
```

Each country is identified either by its name (`country`) and, for 249 of them, by the two and three digits **ISO 3166-1** code (respectively `iso2` and `iso3`). `type` indicates whether the entity is the main part of a sovereign country (`"main"`), a part or a dependency of a sovereign country (`"part"` or "`dependency`"), or an indeterminate territory (`"indeterminate"`). `sovereign` is the name of the sovereign country (equal to `country` is the entity is of the `"main"` category). `capital` is the name of the capital of the country and `status` is the United Nation status. The name of the country is also provided in 5 languages in the columns `en`, `fr`, `de`, `es`, `it`.

Countries are grouped in different entities:

- United Nation's regions `region` (Africa, Americas, Asia, Europe, Oceania) which are decomposed in 22 subregions `subregion`,
- World Bank's regions `wbregion` (Antarctica, East Asia & Pacific, Europe & Central Asia, Latin America & North America, South Asia and Sub-Saharan Africa),
- Economic group `economy`,
- Income group `income`.

[3] We use `rmapshaper::ms_simplify` to reduce the size of the file by a factor of about 10, except for small islands which were kept as in the initial file.

Finally, two numeric covariates are provided, the population (`pop`) and the gross domestic product (`gdp`).

`ne_towns` is another `sf` that contains 7342 towns, obtained from a shape file of **Natural Earth** called populated places; it contains their names `name`, the iso codes of the country they belong to (`iso2` and `iso3`), a boolean indicating whether it is a capital (`capital`) the population `pop` and the point coordinates `point`.

```
ne_towns %>% print(n = 2)
#> Simple feature collection with 7342 features and 5 fields
#> Geometry type: POINT
#> Dimension:     XY
#> Bounding box:  xmin: -179.59 ymin: -90 xmax: 179.3833 ymax: 82.48332
#> Geodetic CRS:  WGS 84
#> First 2 features:
#>   iso2 iso3                  name capital   pop                       point
#> 1   UY  URY Colonia del Sacramento   FALSE 21714 POINT (-57.83612 -34.46979)
#> 2   UY  URY              Trinidad   FALSE 21093     POINT (-56.901 -33.544)
```

These two `sf` are exported by `countries` and can therefore be used directly.

Selecting countries

The `countries` function is provide to extract a subset of countries. It's first argument is called `name` and its default value is `NA`. In this case all the countries are returned.

```
countries()
```

`name` is a vector of character that should contain either countries, regions or subregions (but not a mixture). Let's first select one country, for example France:

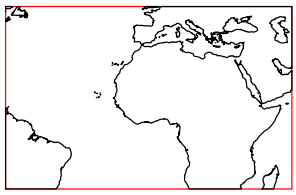
```
countries("France") %>% as_tibble %>% select(1:5)
#> # A tibble: 1 x 5
#>   iso2  iso3  type  country sovereign
#>   <chr> <chr> <fct> <chr>   <chr>
#> 1 FR    FRA   main  France  France
```

by default, only the main part of France is returned. The parts (dependencies) can also be returned by setting the `part` (`dependency`) argument to `TRUE`:

```
fr_parts <- countries("France", part = TRUE)
fr_parts %>% pull(country)
#> [1] "France"     "Guadeloupe" "Guyane"     "Martinique" "Mayotte"
#> [6] "Reunion"
countries("France", dependency = TRUE) %>% pull(country)
#> [1] "Clipperton Island"                    "France"
#> [3] "French Polynesia"                     "French Southern and Antarctic Lands"
#> [5] "New Caledonia"                        "Saint Barthelemy"
#> [7] "Saint Martin"                         "Saint Pierre and Miquelon"
#> [9] "Wallis and Futuna"
```

`countries` returns an object of class `countries` that inherits from `sf`. It has a `type` attribute that is either `country`, `region`, `subregion` or `world`, depending on the kind of entities selected in the `name` argument. It also have a `bb` attribute which contains the bounding box and, if `coastlines` is `TRUE` (the default), a `bg` attribute that is a `sfc` containing the coastlines in the bounding box of the selected territories. For example, for France with the parts selected, we have:

```
fr_parts %>% attr("bb") %>% plot(border = "red")
fr_parts %>% attr("bg") %>% plot(add = TRUE)
```

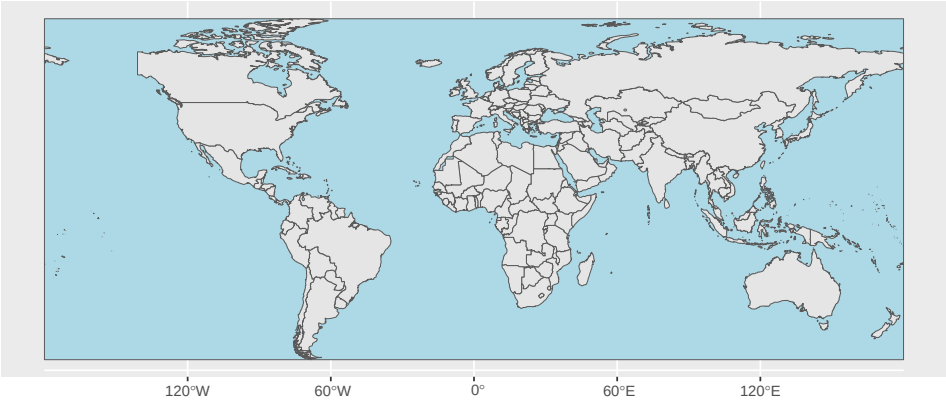


Note that the bounding box for France extends on the south-east to La Reunion and on the west to Martinique and Guadeloupe. If a region or a subregion is selected, the `part` and `dependency` arguments reason on the region / subregion and not on the countries that belongs to the region / subregion. For example, France is part of the Western Europ subregion. Selecting this subregion with `part = TRUE`:

```
countries("Western Europe", part = TRUE) %>% pull(country)
#> [1] "Austria"       "Belgium"       "France"        "Germany"
#> [5] "Liechtenstein" "Luxembourg"    "Monaco"        "Netherlands"
#> [9] "Switzerland"
```

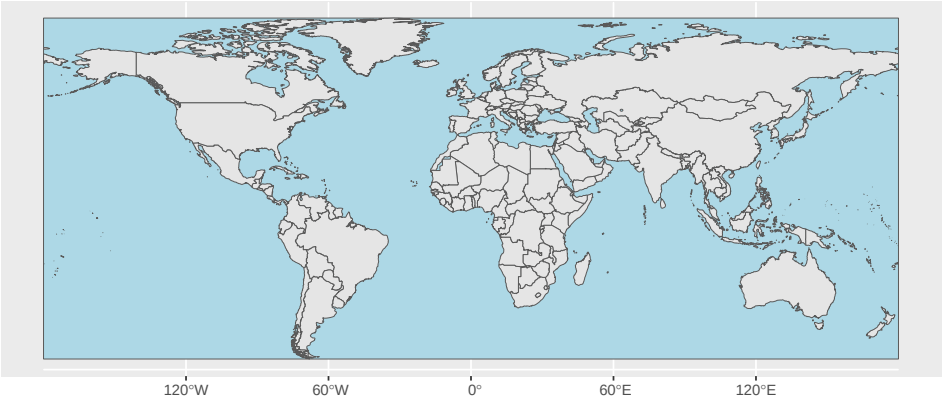
doesn't return the overseas departments which are parts of France but which are not part of the subregion France belongs to. A `indeterminate` argument, by default equal to `FALSE` enables to select or not the indeterminate territories. The `exclude` and `include` arguments are characters which enables to exclude or include entities from the set selected by the `name` argument. For example, to get a world map without the Antarctica:

```
countries(exclude = "Antarctica", coastlines = FALSE) %>% plot
```



This is a world map with sovereign countries, without parts and dependencies. To add Alaska (a part of the USA) and Greenland (a dependency of Denmark), we use the `include` argument:

```
countries(exclude = "Antarctica", coastlines = FALSE,
          include = c("Alaska", "Greenland")) %>% plot
```



Note the use of the `plot` method for `countries` object. A basic plot is obtained without argument, we'll develop further the use of this method latter. Note for now that, contrary to `sf`, the `plot` method for `countries` object use **ggplot2**, more specifically it is obtained by a call to `ggplot(x) + geom_sf()`.

Selecting towns

`necountries::towns` function select towns with a first mandatory argument that can be either a character vector (as in `necountries::countries`) or a `countries` object:

```
we <- countries(c("France", "Spain"))
towns(we) %>% print(n = 2)
#> Simple feature collection with 104 features and 5 fields
#> Geometry type: POINT
#> Dimension:     XY
#> Bounding box:  xmin: -8.729994 ymin: 35.3 xmax: 9.450007 ymax: 50.95042
#> Geodetic CRS:  WGS 84
#> First 2 features:
#>   iso2 iso3     name capital    pop                     point
#> 1   ES  ESP   Mérida   FALSE  52423  POINT (-6.337998 38.912)
#> 2   ES  ESP Marbella   FALSE 186131 POINT (-4.88333 36.51662)
```

is equivalent to:

```
towns(c("France", "Spain"))
```

`capital` is a boolean that unsures that the capital of every countries are selected. `size` is a numeric that indicates the minimum population of the towns to be returned. The default

value of `size` is 0 so that all the towns of the selected countries are returned. Consider for example Australia:

```
towns("Australia", size = 2E06)
#> Simple feature collection with 2 features and 5 fields
#> Geometry type: POINT
#> Dimension:     XY
#> Bounding box:  xmin: 144.9731 ymin: -37.81809 xmax: 151.2125 ymax: -33.87137
#> Geodetic CRS:  WGS 84
#>   iso2 iso3      name capital     pop                       point
#> 1   AU  AUS Melbourne   FALSE 4170000 POINT (144.9731 -37.81809)
#> 2   AU  AUS    Sydney   FALSE 4630000 POINT (151.2125 -33.87137)
```

returns the two largest towns of Australia (Melbourne and Sydney). Setting `capital` to `TRUE`, we get:

```
towns("Australia", size = 2E06, capital = TRUE)
#> Simple feature collection with 3 features and 5 fields
#> Geometry type: POINT
#> Dimension:     XY
#> Bounding box:  xmin: 144.9731 ymin: -37.81809 xmax: 151.2125 ymax: -33.87137
#> Geodetic CRS:  WGS 84
#>   iso2 iso3      name capital     pop                       point
#> 1   AU  AUS  Canberra    TRUE  327700  POINT (149.129 -35.28303)
#> 2   AU  AUS Melbourne   FALSE 4170000 POINT (144.9731 -37.81809)
#> 3   AU  AUS    Sydney   FALSE 4630000 POINT (151.2125 -33.87137)
```

ie, Canberra, which is only 327,700 inhabitants, is returned because it is the capital of Australia.

Towns can be selected within the `necountries::countries` function by using the `capital` and the `towns` arguments. `capital` has exactly the same meaning as previously, `towns` can either be a boolean (`FALSE` no towns, except eventually the capital are returned or `TRUE`, all the towns are returned) or a numeric which is passed to the `size` argument of `necountries::towns`.

```
aus <- countries("Australia", towns = 2E06, capital = TRUE)
```

When arguments `capital` and/or `towns` are used in `necountries::countries`, the returned `countries` object has a `towns` attribute that contains the towns:

```
attr(aus, "towns")
#> Simple feature collection with 3 features and 5 fields
```

```
#> Geometry type: POINT
#> Dimension:     XY
#> Bounding box:  xmin: 144.9731 ymin: -37.81809 xmax: 151.2125 ymax: -33.87137
#> Geodetic CRS:  WGS 84
#>   iso2 iso3      name capital     pop                       point
#> 1   AU  AUS  Canberra    TRUE  327700  POINT (149.129 -35.28303)
#> 2   AU  AUS Melbourne   FALSE 4170000 POINT (144.9731 -37.81809)
#> 3   AU  AUS    Sydney   FALSE 4630000 POINT (151.2125 -33.87137)
```

A `labels` method for `countries`'s objects is provided. It returns a `sf` with a `POINT` `sfc` which contains the entities that can be labelled, which can be either countries, capitals, towns or a combination of them. For example:

```
countries(c("Portugal", "Spain"), towns = 1E06, capital = TRUE) %>%
    labels(var = c("country", "towns", "capital"))
#> Simple feature collection with 7 features and 2 fields
#> Geometry type: POINT
#> Dimension:     XY
#> Bounding box:  xmin: -9.146812 ymin: 37.40502 xmax: 2.181424 ymax: 41.38525
#> Geodetic CRS:  WGS 84
#> # A tibble: 7 x 3
#>   type    name                    point
#>   <chr>   <chr>               <POINT [°]>
#> 1 country Portugal  (-8.289959 39.53647)
#> 2 country Spain      (-3.44966 39.94038)
#> 3 capital Lisbon    (-9.146812 38.72467)
#> 4 capital Madrid    (-3.685297 40.40197)
#> 5 town    Porto     (-8.621947 41.15195)
#> 6 town    Seville   (-5.980007 37.40502)
#> 7 town    Barcelona  (2.181424 41.38525)
```

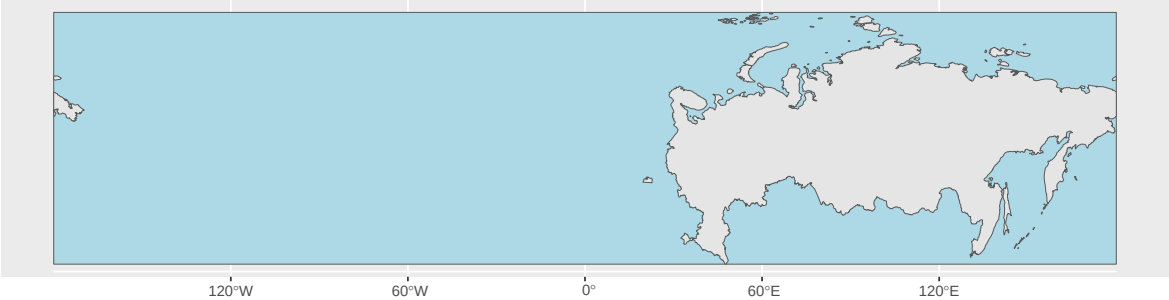
The `type` and `name` columns contains entities' types (country, capital or town) and names. The geometry is the coordinates of the cities for `capital` and `town` entities and the point obtained by the `sf::st_point_on_surface` function for countries. We'll see later that this function can be used to display labels on maps.

Coordinate reference system

natural earth use geographical coordinates with the **WGS84** datum, the well known **crs** `+proj=longlat +datum=WGS84 +no_defs` with **epsg** code 4326.

A common problem is that some territories are on both sides of the 180 E/W longitude. In this case, the `shift` argument can be used, so that the center of the world map is the 180 longitude and not the Greenwich (0) longitude (internally, the `sf::st_shift_longitude` function is used). Consider for example Russia:

```
countries("Russia", coastlines = FALSE) %>% plot
```



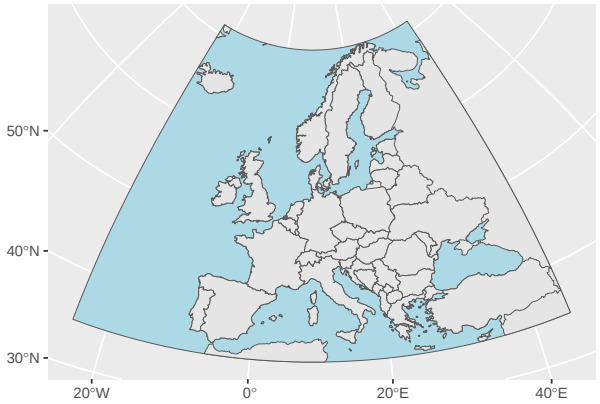
A small part of Russia appears on the left hand size of the map. Using `shift = TRUE`, we get:

```
countries("Russia", coastlines = FALSE, shift = TRUE) %>% plot
```

To use projected **crs**, either the `utm` or the `crs` arguments can be used. The `crs` argument, if used, is passed to `sf::st_transform` in order to transform the geometry in the required `crs`. The `utm` argument enables the use of the Universal Transverse Mercator coordinate system which divides earth into 60 zones. If `TRUE`, the most relevant zone is automatically selected

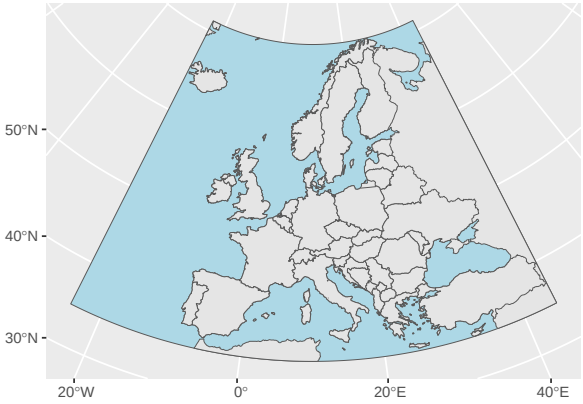
but the user can also select a specific zone using an integer from `0L` to `60L`. For example, to get a utm projected map of Europe without Russia but with Turkey and Cyprus:

```
countries("Europe", utm = TRUE, extend = 1.1,
          include = c("Turkey", "Cyprus", "Northern Cyprus"),
          exclude = "Russia") %>% plot
```



The Lambert conform conic projection for Europe can be used by setting the **crs** argument to the **epsg** code of this **crs** which is 3034:

```
countries("Europe", crs = 3034, extend = 1.5,
          include = c("Turkey", "Cyprus", "Northern Cyprus"),
          exclude = "Russia") %>% plot
```



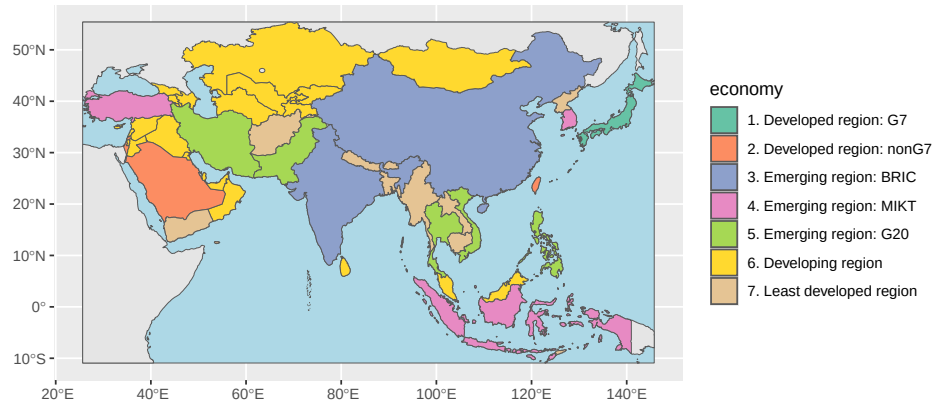
Thematic maps

Until now, we drew very basic maps, in order to show the set of countries selected. More advanced maps can be produced by:

- filling countries using a categorical or a numerical variable,
- maping the shape or the size of points (that can be either the capital or the centroid of countries) with a categorical or a numerical variable,
- adding labels for countries, capitals and/or towns.

For example the `economy` variable is a factor that contains the economic group (developed, emerging, etc.). To fill the different countries with colors associated to the modalities of this factor, we use the `fill` argument:

```
countries("Asia", exclude = "Russia") %>% plot(fill = "economy")
```

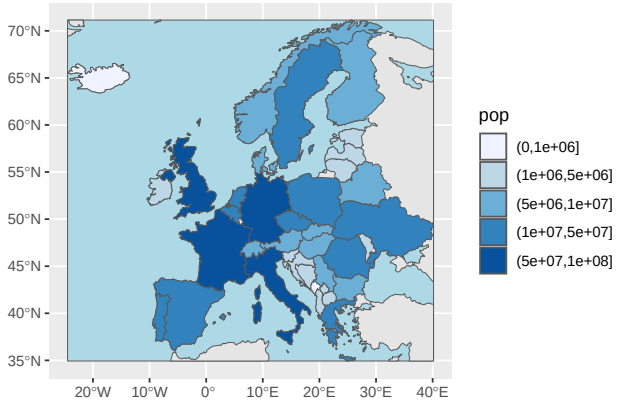


The palettes used are from **ColorBrewer**. Any qualitative palettes can be used using the `palette` argument. For exemple, to use the `"Dark2"` palette:

```
countries(c("Asia"), exclude = "Russia") %>%
    plot(fill = "economy", palette = "Dark2")
```

For numeric variables, bins can be constructed using the `bks` argument. For example, `pop` contains the population of the countries:

```
countries("Europe", exclude = "Russia") %>%
    plot(fill = "pop", bks = c(0, 1E06, 5E06, 1E07, 5E07, 1E08, Inf))
```

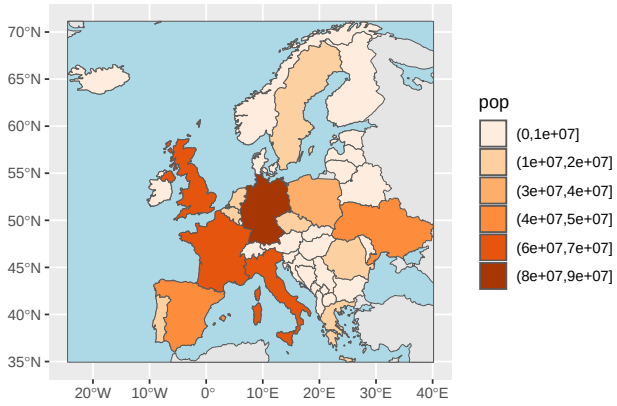


By default, the `"Blues"` palette is used. As previously, any sequential or divergent palette can be used using the `palette` argument. For example, to use the `"PuBu"` divergent palette:

```
countries("Europe", exclude = "Russia") %>%
    plot(fill = "pop", bks = c(0, 1E06, 5E06, 1E07, 5E07, 1E08, Inf),
         palette = "PuOr")
```

The bins can also automatically be computed using the **classInt** package. For this purpose, the `plot` method have `style` and `n` arguments that are passed to `classInt::classIntervals`:

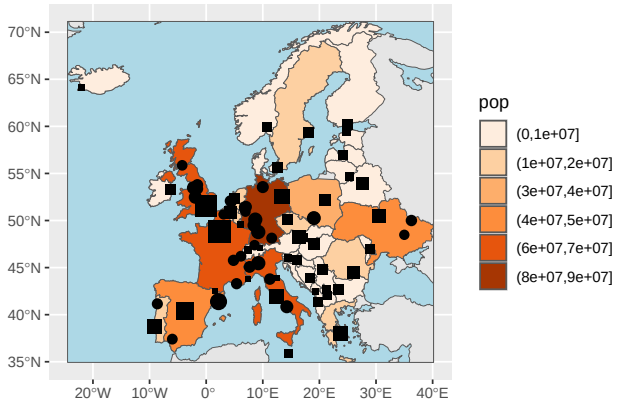
```
countries("Europe", exclude = "Russia") %>%
    plot(fill = "pop", n = 10, style = "pretty",
         palette = "Oranges")
```



Points can be added to the map if the `countries` includes a `"towns"` attribute, which is the case if `capital` and `towns` are not `FALSE` in the call to the `countries` function. If `capital =`

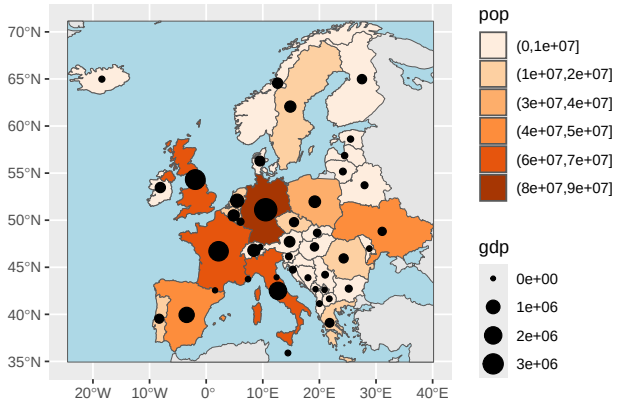
`TRUE` and `towns = FALSE`, the capitals are represented by a point with a size related to their populations. If `towns` is not `FALSE`, some non-capital towns are added, with a special shape and a size related to their population:

```
countries("Europe", exclude = "Russia", capital = TRUE, towns = 1E06) %>%
    plot(fill = "pop", n = 10, style = "pretty",
         palette = "Oranges")
```



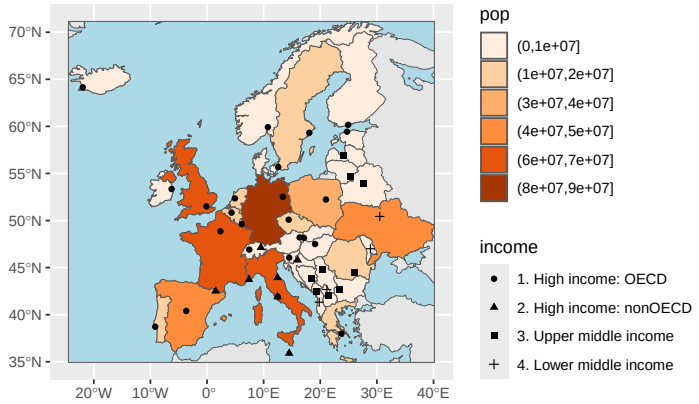
One point for each country (which can be either the capital or the centroid of the country) can also be associated to a numeric or a categorical variable. For example, we can map the point size of the centroid to the gross domestic product using the `centroid` argument:

```
countries("Europe", exclude = "Russia") %>%
    plot(fill = "pop", centroid = "gdp", n = 10, style = "pretty",
         palette = "Oranges")
```



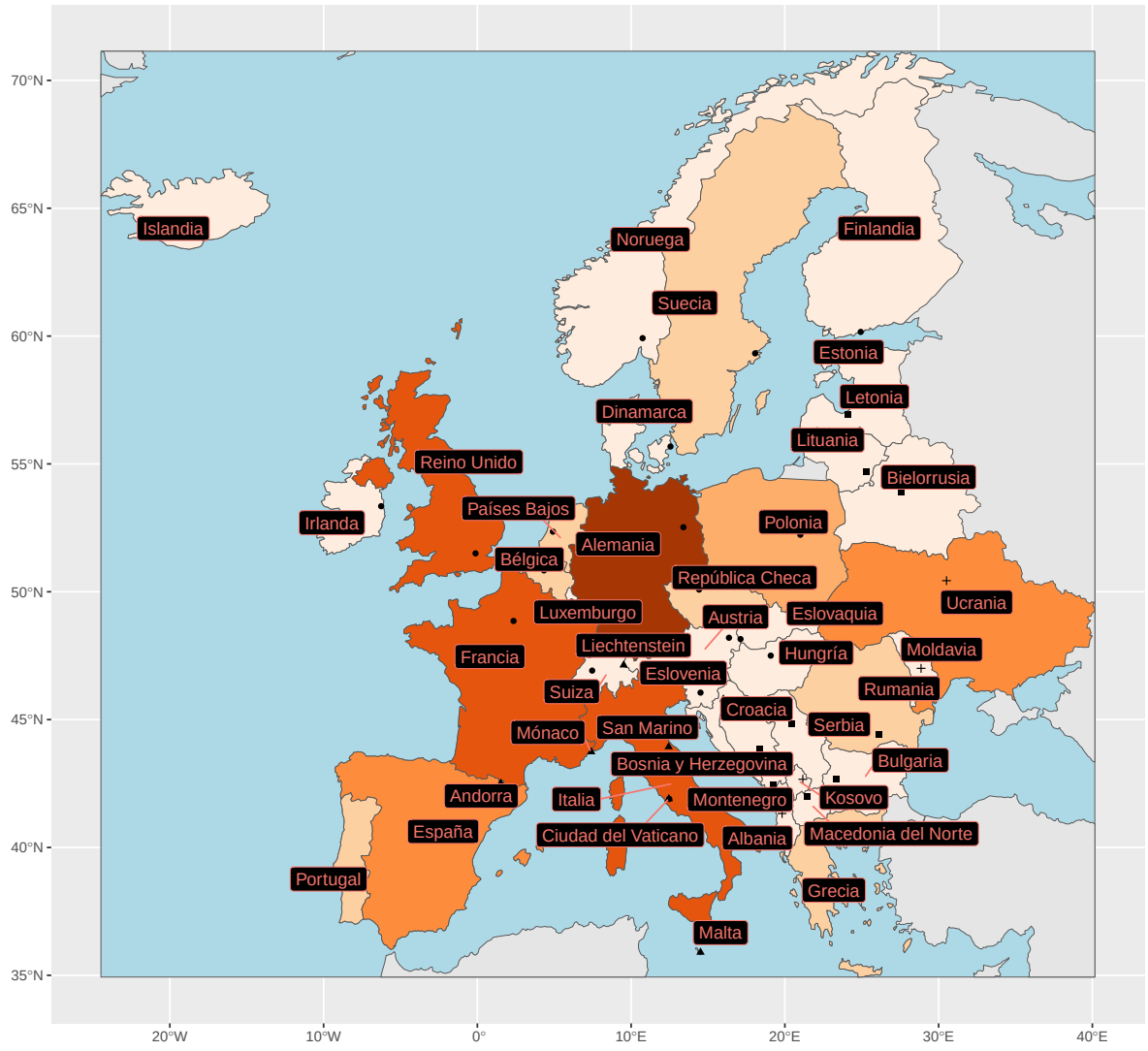
and we can perform the same operation (this time for a categorical variable `income`) for the points defined by the position of the capital using the `capital` argument, so that the shape of the points is maped to the categorical variable.

```
countries("Europe", exclude = "Russia", capital = TRUE) %>%
    plot(fill = "pop", capital = "income", n = 10, style = "pretty",
         palette = "Oranges")
```



Labels can be added using the `labels` argument, which is a character that countains any combination of `"country"`, `"capital"` and `"towns"`. Moreover, for the names of the countries, 5 different languages are available. Here, we use the `lang` argument of `necountries::countries` to display countries' names in Spanish:

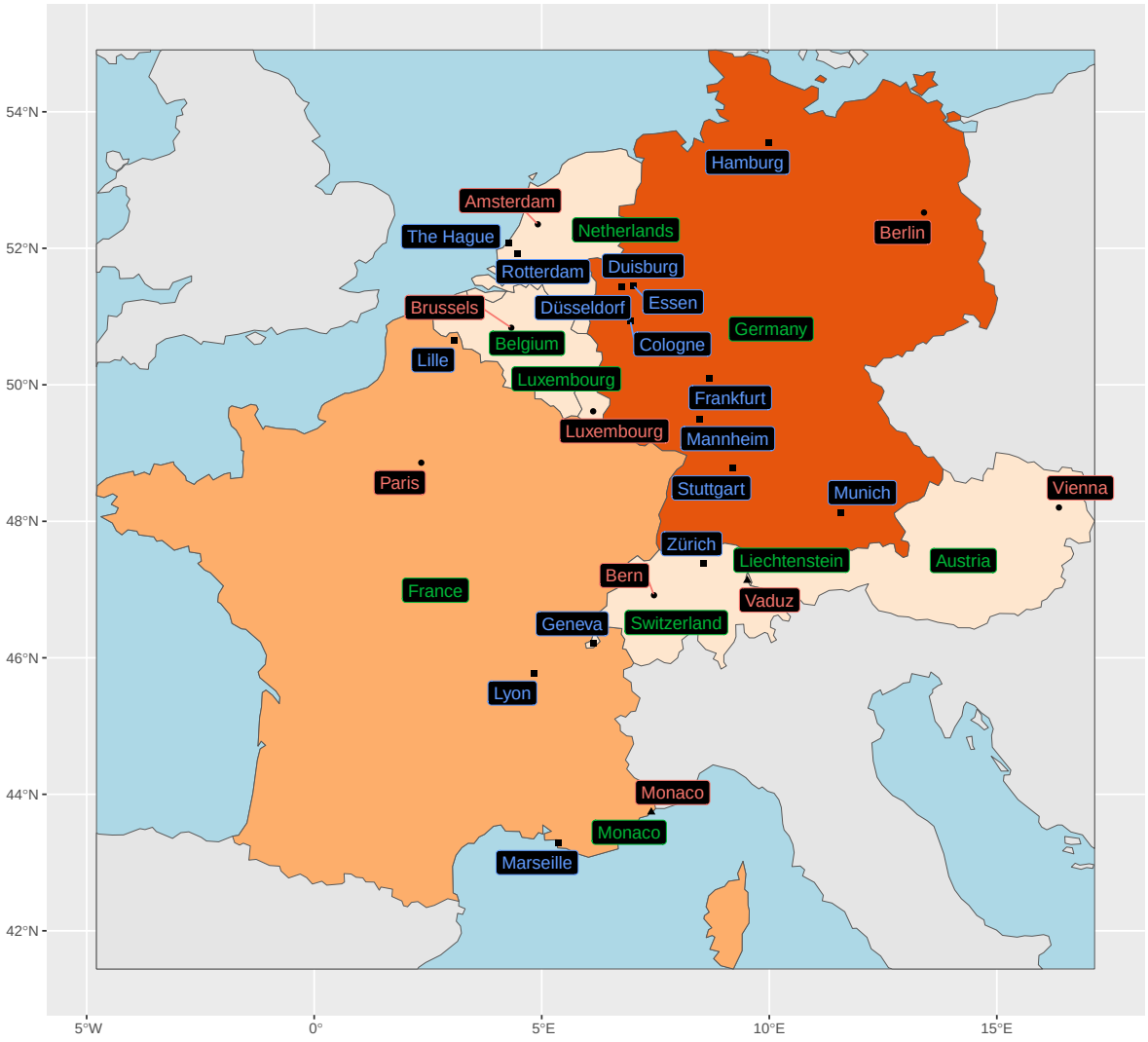
```
countries("Europe", exclude = "Russia", capital = TRUE, lang = "es") %>%
    plot(fill = "pop", capital = "income", n = 10, style = "pretty",
         palette = "Oranges", labels = "country") +
    labs(x = NULL, y = NULL) +
    guides(fill = "none", shape = "none")
```



In a small territory, we can also add labels for towns and capitals. the **ggrepel** package is used to avoid overplotting of labels. In the next map, we plot the countries of Western Europe with their capitals and towns of more than a million inhabitants:

```
countries("Western Europe", capital = TRUE, towns = 1E06) %>%
    plot(fill = "pop", capital = "income", n = 4, style = "pretty",
         palette = "Oranges", labels = c("country", "capital", "towns")) +
```

```
    labs(x = NULL, y = NULL) +
    guides(fill = "none", shape = "none")
```



External data

The `sf` returned by `countries` can also be joined with an external tibble. For this purpose, a `left_join` method for `countries`'s objects is provided. The external tibble must have a column that identifies the entity, which can be either the name or the 2 or 3 digits iso code. To illustrate this feature, we consider two real world examples. The first tibble is called `slave_trade` and is used in Nunn (2008). In this article, the long-term effects of slave trade on African countries' economic activity is analysed. The data set contains 52 African countries and the main covariate is the number of slaves exported from each country divided by the average population during the slave trade period. We compute this covariate and select some existing columns, `gdp` (gdp per capita in 2000) and `colony` (a factor containing the previous colonizator):

```
slave_trade <- slave_trade %>%
    mutate(slaves = slaves / pop) %>%
    select(country, slaves, gdp, colony)
```

The `left_join` method takes only three arguments: the `countries` object, the external tibble and the column that countains countries' identifiers. A `check_join` function is provided, with a further argument called `side`, that checks:

- whether all the countries of the tibble are present in the `countries` object (`side = "right"`, the default),
- whether all the countries of the `countries` object are present on the tibble (`side = "left"`),
- whether the two sets of countries are the same (`side = "both"`).

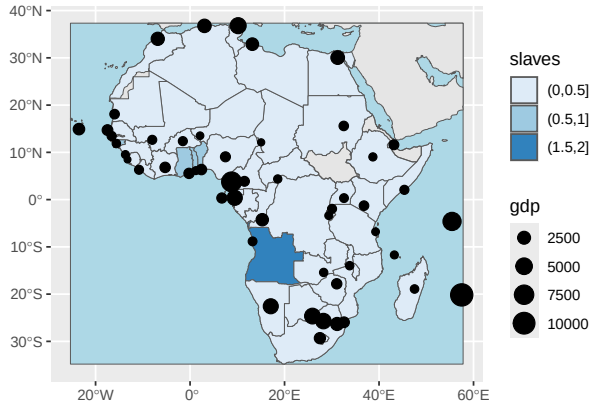
```
countries("Africa") %>%
    check_join(slave_trade, by = "country", side = "both")
#>
#> Countries in the external tibble not in the countries' sf:
#>  Cape Verde Islands, Sao Tome & Principe, Swaziland, Democratic Republic of Congo
#>
#> Countries in the countries' sf not in the external tibble:
#>  Cabo Verde, D.R. Congo, Eritrea, Sao Tome and Principe, Somaliland, South Sudan,
#> eSwatini
```

4 countries from the `slave_trade` tibble don't have correspondance with the `countries` object because of different spellings. On the contrary, 3 countries of the `countries` object are not present in `slave_trade`: Eritrea and South Sudan (which were not or freshly independent by the time of the article), and Somaliland. We then correct the spelling of the 4 countries in the `slave_trade` tibble before joining:

```
slave_trade <- slave_trade %>%
    mutate(country = case_when(country == "Democratic Republic of Congo" ~ "D.R. Congo",
                               country == "Cape Verde Islands" ~ "Cabo Verde",
                               country == "Sao Tome & Principe" ~ "Sao Tome and Principe",
                               country == "Swaziland" ~ "eSwatini",
                               .default = country))
```

and we can now perform the join:

```
strade <- countries("Africa", capital = TRUE) %>% select(iso2:status, point) %>%
    left_join(slave_trade, "country")
strade %>% plot(fill = "slaves", n = 5, type = "pretty", capital = "gdp")
```



`sp_solow` contains the data used by Ertur and Koch (2007) to estimate a Solow's growth model with externalities and taking into account spatial dependence.

```
sp_solow %>% print(n = 3)
#> # A tibble: 91 x 6
#>   name      code   gdp60  gdp95 saving labgwth
#>   <chr>     <chr>  <dbl>  <dbl>  <dbl>   <dbl>
#> 1 Angola    AGO    5136.  2629. 0.0736  0.0233
#> 2 Argentina ARG   18733. 24738. 0.178   0.0165
#> 3 Australia AUS   26480. 45331. 0.247   0.0210
#> # i 88 more rows
```

The values of the gross domestic product are given for the years 1960 and 1995, we compute the average growth rate for the period:

```
sp_solow <- sp_solow %>%
    mutate(growth = (gdp95 / gdp60) ^ (1 / 35) - 1)
```

Either `name` or `code` can be used to joint `sp_solow` with the `countries`' object. It is much safer to use `code` as it avoids the problem of small differences in countries' names. As a lot of countries of the world are not present in `sp_solow` (especially most of the communist countries), we just check whether all the the countries of `sp_solow` are present in the `countries` object:

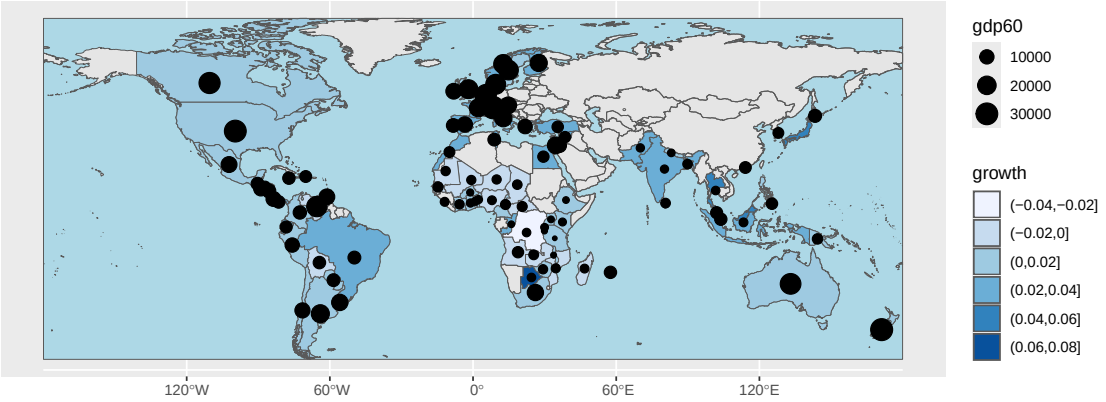
```
countries() %>% check_join(sp_solow, by = "code")
#>
#> Countries in the external tibble not in the countries' sf:
#>  HKG, ZAR
```

The two problems is that the D.R. Congo (`iso3` code `COD`) used to be called Zaire (`iso3` code `ZAR`) and that Hong Kong, which is a part of a China in `ne_countries` was considered as a sovereign country by the time of the study.

```
sp_solow <- sp_solow %>% mutate(code = ifelse(code == "ZAR", "COD", code))
sps <- countries(include = "Hong Kong", exclude = "Antarctica") %>%
    select(iso2:status, point) %>%
    left_join(sp_solow, by = "code")
```

We then draw a world map with the color of the countries related to the annual growth during the 1960-95 period and a point with a size related to the initial (1960) gdp.

```
sps %>% plot(fill = "growth", centroid = "gdp60")
```



Details of the computation

The most aggregate file is called **sovereignty** and contains 209 entities: the 193 UN countries and 16 more territories characterized as follow:

- **disputed**: Kosovo,
- **sovereign country**: Northern Cyprus, Somaliland, Taiwan, Vatican
- **indeterminate**: Antarctica, Bajo Nuevo Bank (Petrel Is.) (small inhabited islands Colombia / United States of America / Nicaragua and Jamaica), Bir Tawil (Egypt / Sudan), Brazilian Island (river islands Brazil / Uruguay), Cyprus No Mans Area, Scarborough Reef (islands China / Philippines / Taiwan), Serranilla Bank (caribbean islands Colombia / Nicaragua), Siachen Glacier (India / Pakistan), Southern Patagonian Ice Field (Argentina / Chile), Spratly Islands (Brunei / China / Malaisia / Philippines / Taiwan / Viet Nam), Western Sahara (Morocco / Sahrawi Arab Democratic Republic)

Four more sovereign countries but with not full recognization are added: Kosovo, Northern Cyprus, Somaliland and Taiwan. Note that Palestine, one of the two observer states is not included in this file.

The **countries** shape file contains 49 more entities that are mostly countries' dependencies. The 258 entities are categorized the following way:

- **sovereign country** : 185 (**181** UN members + Vatican, Somaliland, Taiwan and Northern Cyprus)
- **sovereignty** : **2** (Cuba and Kazakhstan)
- **country** : 19 (including **9** Sovereign countries: Netherlands, France, China, Finland, United Kingdom, United States of America, Australia, New Zealand, Denmark) + Sint Maarten, Hong Kong, Greenland, Curaçao, Aruba, Jersey, Guernsey, Isle of Man, Aland, Macao.
- **disputed** : **1** UN member Israel, Kosovo, Gibraltar, British Indian Ocean Territory, Falkland Islands,
- **indeterminate** : same 11 entities as previously + Palestine
- **lease** : US Naval Base Guantanamo Bay, Baykonur Cosmodrome
- **dependency** : 33 territories

With a enlarged definition of sovereign countries, we then have $181+2+9+1 = 193$ UN members, 2 observers (Vatican, Palestine) and 4 not fully recognized countries (Kosovo, Nothern Cyprus, Somaliland, Taiwan), 199 in total.

The dependency set consists of the 33 territories categorized as such, the remaining 10 territories of the country category, the 3 territories belonging to the disputed category (except Israel and Kosovo) and the 2 territories in the lease category (Guantanamo and Baykonur), 48 in total, presented in the following list, with the name of the sovereign country in bold.

- **United Kingdom**: Dhekelia Sovereign Base Area, Gibraltar, Akrotiri Sovereign Base Area, Turks and Caicos Islands, Pitcairn Islands, Montserrat, Anguilla, British Virgin Islands, Cayman Islands, Bermuda, Saint Helena, Jersey, Guernsey, Isle of Man, British Indian Ocean Territory, South Georgia and the Islands, Falkland Islands
- **France**: Saint Martin, New Caledonia, Saint Pierre and Miquelon, French Polynesia, French Southern and Antarctic Lands, Saint Barthelemy, Wallis and Futuna, Clipperton Island
- **Netherlands**: Sint Maarten, Curaçao, Aruba
- **Cuba**: US Naval Base Guantanamo Bay
- **China**: Hong Kong S.A.R., Macao S.A.R
- **Kazakhstan**: Baykonur Cosmodrome
- **Denmark**: Greenland, Faroe Islands
- **United States of America**: United States Minor Outlying Islands, United States Virgin Islands, Puerto Rico, American Samoa, Guam, Northern Mariana Islands
- **Australia**: Heard Island and McDonald Islands, Indian Ocean Territories, Norfolk Island, Coral Sea Islands, Ashmore and Cartier Islands
- **Sao Tome and Principe**: Sao Tome and Principe
- **Finland**: Aland
- **New Zealand**: Cook Islands, Niue

The 11 remaining territories correspond to the Indeterminate category.

As stated previously, there is a need to split some entities in different features. The complete list of the 39 part territories is given below:

- **France (5)**: Reunion, Mayotte, Guyane, Martinique, Guadeloupe (the 5 overseas deparments),
- **United States of America (2)**: Alaska and Haiwaii
- **Norway (2)**: Bouvet (a small Island in Antarctica) and Svalbard and Jan Mayen (a set of Islands far north of the Norwegian coasts),
- **Netherlands (1)**: Bonaire, Sint Eustatius and Saba (Caribbean Islands)
- **Portugal (2)**: Azores and Madeira
- **Spain (1)**: Cannaries
- **Mauritius (2)**: Rodrigues, Agalega
- **New Zealand (6)**: Tokelau, Chatman, Kermadec, Auckland, Campbelle, Antipodes
- **Chile (1)**: Easter Island
- **Colombia (1)**: San Andres, Providencia and Santa Catalina
- **South Africa (1)**: Prince Edward Islands
- **Ecuador (1)**: Galapagos
- **Australia (1)**: Macquarie Island
- **Denmark (1)**: Bornholm
- **Equatorial Guinea (1)**: Annobon
- **Seychelles (6)**: Aldabra, Coetivy, Alphonse, Attol Saint-Joseph, Platte, denis
- **Antigua and Barbuda (1)**: Redonda

- **Indian Ocean Territories (2)**: Christmas Islands, Cocos Islands
- **United States Minor outlying Islands (2)**: Navassa (Navassa is a small caribbean islands as the other ones are in the Pacific)

Some very small islands from Japan, Brazil, the United Kingdom and Venezuela were removed.

Note that the Indian Ocean Territories and the United States Minor outlying Islands are splited in two. We then have 295 entities, 258 from the **countries** file minus 2 (Indian Ocean Territories and US minor outlying islands) plus 39, grouped in 4 categories, as described in the first section of this document.

References

Bivand, Roger. 2023. *classInt: Choose Univariate Class Intervals*. https://CRAN.R-project.org/package=classInt.

Ertur, Cem, and Wilfried Koch. 2007. "Growth, Technological Interdependence and Spatial Externalities: Theory and Evidence." *Journal of Applied Econometrics* 22 (6): 1033–62. https://doi.org/https://doi.org/10.1002/jae.963.

Massicotte, Philippe, and Andy South. 2023. *Rnaturalearth: World Map Data from Natural Earth*. https://CRAN.R-project.org/package=rnaturalearth.

Nunn, Nathan. 2008. "The Long-Term Effects of Africa's Slave Trades." *The Quarterly Journal of Economics* 123 (1): 139–76. https://www.jstor.org/stable/25098896.

Pebesma, Edzer. 2018. "Simple Features for R: Standardized Support for Spatial Vector Data." *The R Journal* 10 (1): 439–46. https://doi.org/10.32614/RJ-2018-009.

Slowikowski, Kamil. 2021. *Ggrepel: Automatically Position Non-Overlapping Text Labels with 'Ggplot2'*. https://CRAN.R-project.org/package=ggrepel.

Wickham, Hadley. 2016. *Ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. https://ggplot2.tidyverse.org.