

GADMTools - Manipulating Shapefiles

jean.pierre.decorps@gmail.com

2021-08-04

What is GADM?

GADM, the Database of Global Administrative Areas, is a high-resolution database of country administrative areas, with a goal of “all countries, at all levels, at any time period. The database has a few export formats, including shapefiles that are used in most common GIS applications.[2] Files formatted for the programming language R are also available, allowing the easy creation of descriptive data plots that include geographical maps. Although it is a public database, GADM has a higher spatial resolution than other free databases and also higher than commercial software such as ArcGIS. GADM is not freely available for commercial use. The GADM project created the spatial data for many countries from spatial databases provided by national governments, NGO, and/or from maps and lists of names available on the Internet (e.g. from Wikipedia).

The GADM website and data repository is hosted at UC Davis in the Hijmans Lab. The Hijman lab is run by Robert Hijmans an Environmental Science and Policy faculty member in the Geography Graduate Group. [source Wikipedia - <https://en.wikipedia.org/wiki/GADM>]

What is GADMTools?

GADMTools is an R package to manipulate shapefiles from GADM and to make geo-statistical representations easily.

GADMTools can use 2 shapefile formats, *SpatialPolygonsDataFrame (SP)* and *Simple Features (SF)*, both provided by GADM as .rds files.

NB: the SF format is supported only from version 3.5 of GADMTools.

Manipulating shapefiles

functions

SpatialPolygons	Simple Features	Description
<code>gadm_sp_loadCountries</code>	<code>gadm_sf_loadCountries</code> <code>gadm_sf_import_shp</code>	downloads or loads one or more shapefiles load a .shp file and convert it to <code>gadm_sf</code> object
<code>gadm_crop</code>	<code>gadm_crop</code>	crop a region to a specific rectangle
<code>gadm_exportToShapefile</code>	<code>gadm_exportToShapefile</code>	Export to ESRI Shapefile
<code>gadm_getBackground</code>	<code>gadm_getBackground</code>	Gets tiles with 'rosm' from OpenStreetMap
<code>gadm_getBbox</code>	<code>gadm_getBbox</code>	get the bounding box of the map
<code>gadm_loadStripped</code>		Load a GADM stripped shapefile
<code>gadm_longTo360</code>	<code>gadm_longTo360</code>	Converts longitudes from -180° - 0° - 180° to 0° - 360°
<code>gadm_remove</code>	<code>gadm_remove</code>	Removes one or more regions from a map in a GADMWrapper/GT2 object
<code>gadm_removeBackground</code>	<code>gadm_removeBackground</code>	Removes the background of a map
<code>gadm_saveStripped</code>		Save a stripped GADM object
<code>gadm_subset</code>	<code>gadm_subset</code>	Extract regions. "subset" does not work since release 3.5-1
<code>gadm_union</code>	<code>gadm_union</code>	Merges regions
<code>listNames</code>	<code>listNames</code>	List the region names for an administrative level
<code>saveAs</code>	<code>saveAs</code>	Save your own GADM shapefile as a .rds file
<code>stripSP</code>		Strip a GADMWrapper object

CAUTION: Functions whose names were previously prefixed by "gadm." are now prefixed by "gadm_" for compliance with the R language coding conventions. Older functions are still available for this release but will be removed in the next release. Generally all the "." in the function names have been replaced by "_".

Function *gadm.loadCountries* has been removed.

Format SP : `gadm_sp_loadCountries()`

This is a main function of GADMTools, with it, you can load or download one or more shapefiles. If you load many shapefiles, the function assembles the shapefiles into one.

The old function `gadm_loadCountries` has been removed.

```
gadm_sp_loadCountries(  
    fileNames,  
  
    level = 0,  
  
    basefile=GADM_BASE,  
  
    baseurl=GADM_URL,  
  
    simplify=NULL  
)
```

Parameter	Description
fileNames	Character vector of named regions. An ISO-3166-1 code or a custom name. You don't have to specify the suffix (admX) nor the file extension (.rds).
level	Integer - the level of the administrative boundaries (0 is the country, higher values equal finer divisions)
basefile	Character - the path of the directory where shapefiles are stored. Default is <code>"/GADM"</code>
baseurl	Character - the url of GADM files. Default is <code>http://biogeo.ucdavis.edu/data/gadm_.8/rds/</code>
simplify	Numeric numerical tolerance value to be used by the Douglas-Peucker algorithm. Higher values use less polygon points (and less memory) and lower values use more polygon points (and more memory). We suggest not going higher than 0.025 in order for intra-country boundaries to align.

Return: Object `gadm_sp`

Format SF : `gadm_sf_loadCountries()`

This is a main function of GADMTTools, with it, you can load or download one or more shapefiles. If you load many shapefiles, the function assembles the shapefiles into one.

```
gadm_sf_loadCountries(  
    fileNames,  
  
    level = 0,  
  
    basefile=GADM_BASE,  
  
    baseurl=GADM_URL,  
  
    simplify=NULL  
)
```

Parameter	Description
fileNames	Character vector of named regions. An ISO-3166-1 code or a custom name. You don't have to specify the suffix (admX) nor the file extension (.rds).
level	Integer - the level of the administrative boundaries (0 is the country, higher values equal finer divisions)
basefile	Character - the path of the directory where shapefiles are stored. Default is “./GADM”
baseurl	Character - the url of GADM files. Default is http://biogeo.ucdavis.edu/data/gadm_.8/rds/
simplify	Numeric numerical tolerance value to be used by the Douglas-Peuker algorithm. Higher values use less polygon points (and less memory) and lower values use more polygon points (and more memory). We suggest not going higher than 0.025 in order for intra-country boundaries to align.

Return: Object *gadm_sf*

Format SF : `gadm_sf_import_shp()`

Sometimes we need to import shapefiles different from those provided by GADM.org. It is possible to read and import a file in shapefile format (.shp,.dbf,.proj) and put it in `gadm_sf` format for use with GADMTools.

```
gadm_sf_import_shp(  
  dir,  
  name,  
  level,  
  del = NULL,  
  renamed = NULL,  
  keepall = FALSE  
)
```

Parameter	Description
dir	Character path to the directory where .shp file is located (eg. “./”)
name	Character - name of the .shp file without the extension (example: “india”),
level	Integer - the administrative level
del	Character vector - the variables (columns) to be deleted (optional if keepall == FALSE)
renamed	Character vector - the variables to be renamed (eg. the administrative fields in GADM are named NAME_X where X is the level, and the ISO code(3)),
keepall	Boolean if it is FALSE (default), allows to keep only the columns useful for GADMTools.

Return: Object `gadm_sf`

Example

```
map <- gadm_sf_import_shp(dir="./", name = "india", level = 2,  
  del = c("DCODE", "NAME3", "SDCODE"),  
  renamed = c('ISO' = 'COUNTRY',  
    'NAME_0' = 'COUNTRY_LO',  
    'NAME_1' = 'NAME1',  
    'NAME_2' = 'NAME2'),  
  keepall = FALSE  
)  
map$sf$ISO <- "IND"  
map$sf$NAME_0 <- "India"
```

`gadm_exportToShapefile()`

Export a `gadm` object to an ESRI Shapefile. This function create a directory with te name provided as parameter which contains 4 files :

- `name.dbf`
- `name.prj`
- `name.shp`
- `name.shx`

Where “name” is the name provided as parameter. Directory is created in the current working directory

`gadm_exportToShapefile(`

```
    x,  
    name  
)
```

Parameter	Description
x	Character - <code>gadm_sp</code> or <code>gadm_sf</code> Object
name	Character - name given to te shapefile

Loading a country

```
library(GADMTools)

# Loading country border (level=0 [default])
# -----
map <- gadm_sf_loadCountries("FRA", basefile = "./")
gadm_plot(map) + theme_light()
```

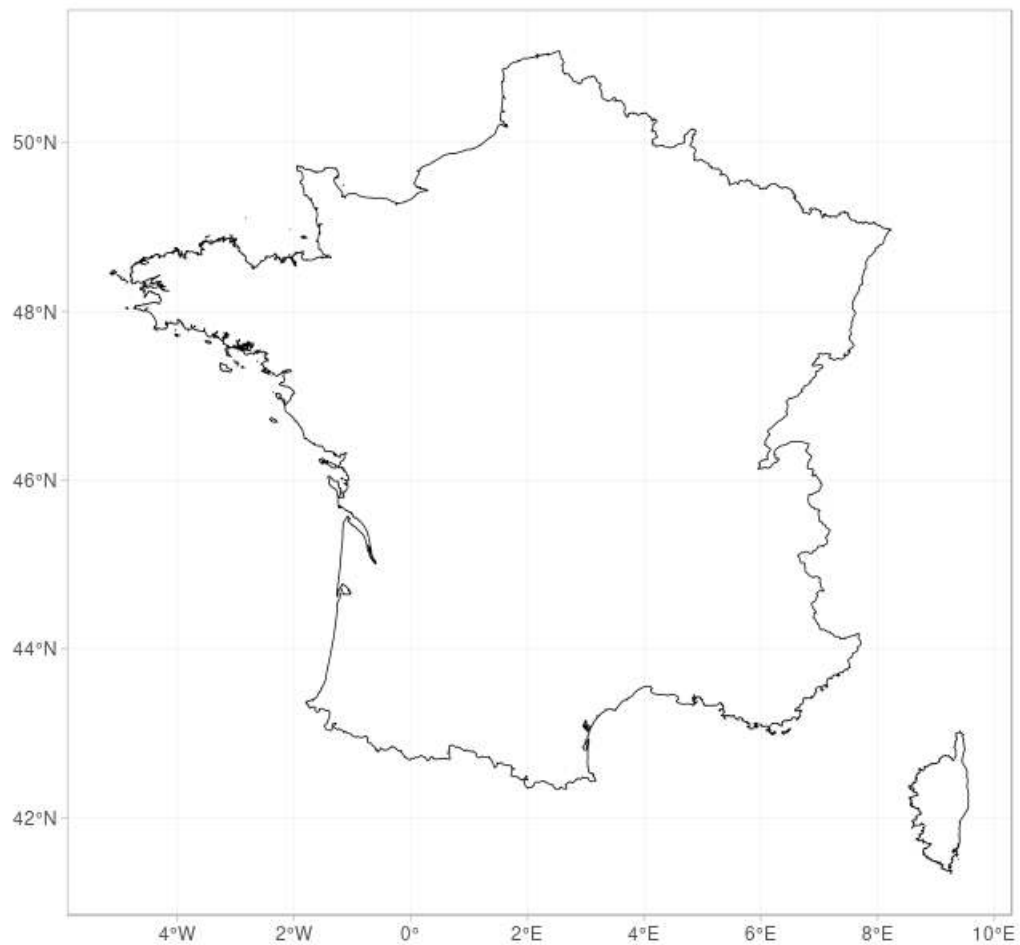


Figure 1: Loading a single country (level = 0)

Loading a country at an administrative level

```
library(GADMTools)
data("Corsica")

# Loading regions @ level = 2]
# -----
map <- gadm_sp_loadCountries(c("FRA"), level=2, basefile = "./")
gadm_plot(map)
```

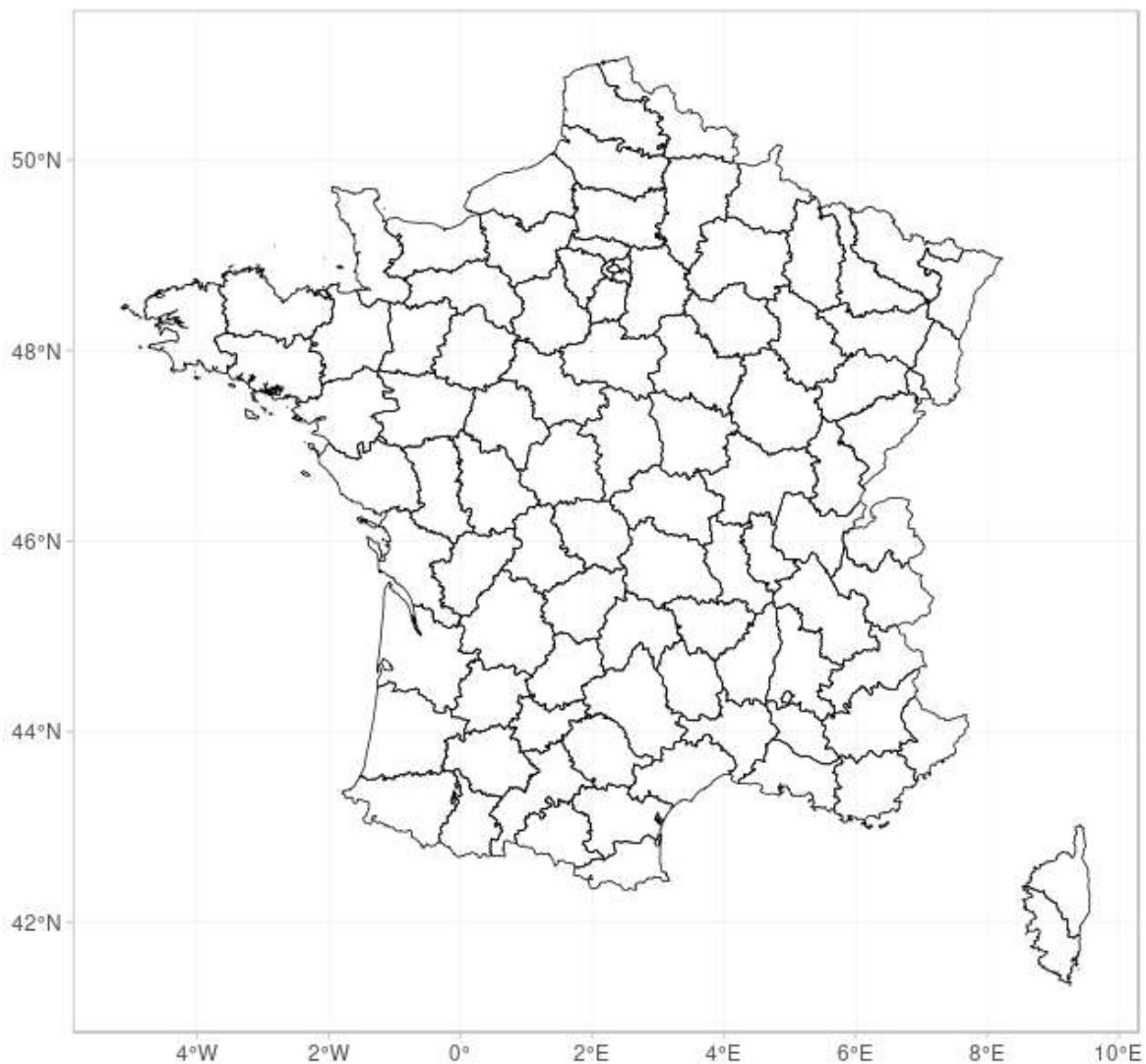


Figure 2: loading regions of a country @ level = 2

NB: you can use `gadm_sf_loadCountries` instead of `gadm_sp_loadCountries`

Assembling many countries

```
library(GADMTools)

# Assemble administrative boundaries (country level = 0)
# -----
map <- gadm_sp_loadCountries(c("BEL","LUX","NLD"), basefile = "./")
gadm_plot(map + theme_light())
```

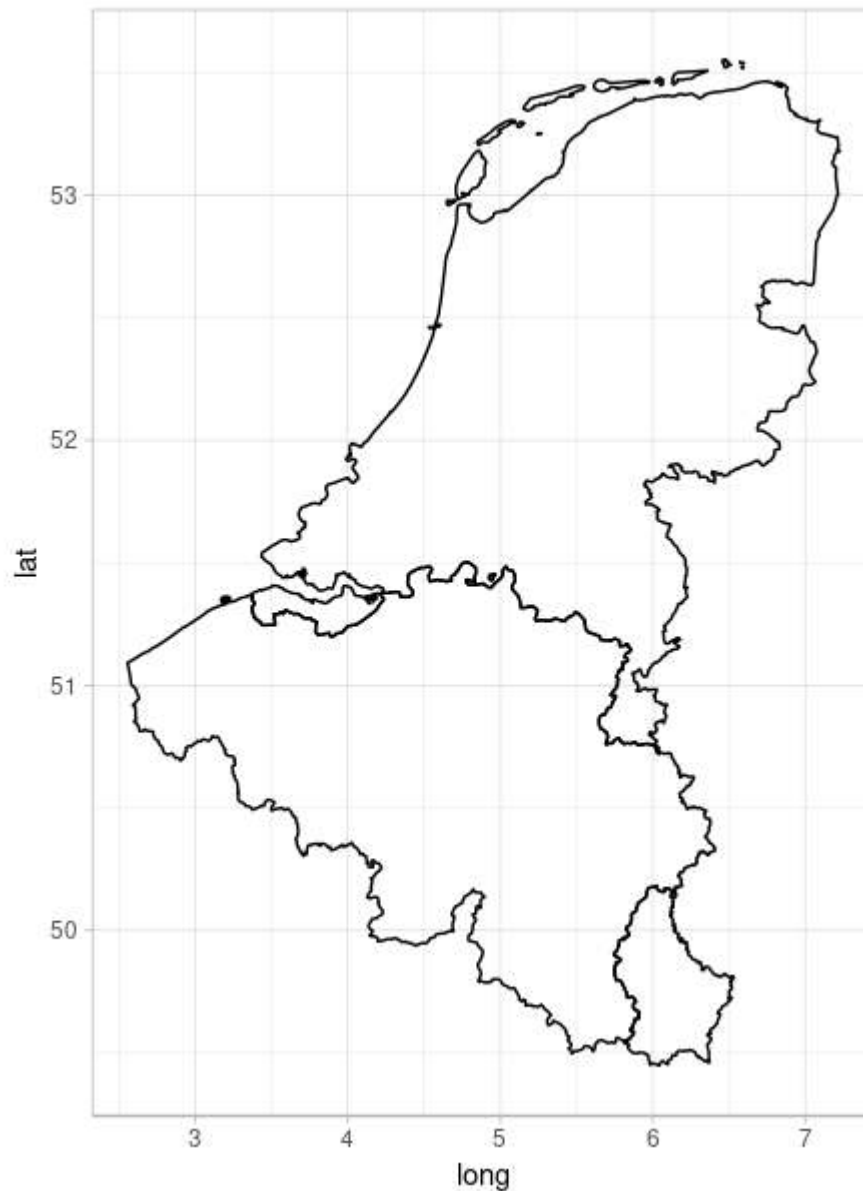


Figure 3: Benelux = Belgium + Luxembourg + Netherlands @ level = 0

NB: you can use `gadm_sf_loadCountries` instead of `gadm_sp_loadCountries`

Extracting regions

```
### First extracting "Corse" from France @ level 4
```

```
FRA <- gadm_sf_loadCountries("FRA", level = 4, basefile = "./")
```

```
Corsica <- gadm_subset(FRA, level=1, regions="Corse")
```

```
gadm_plot(Corsica) %>% gadm_showNorth("t1") %>% gadm_showScale('b1')
```

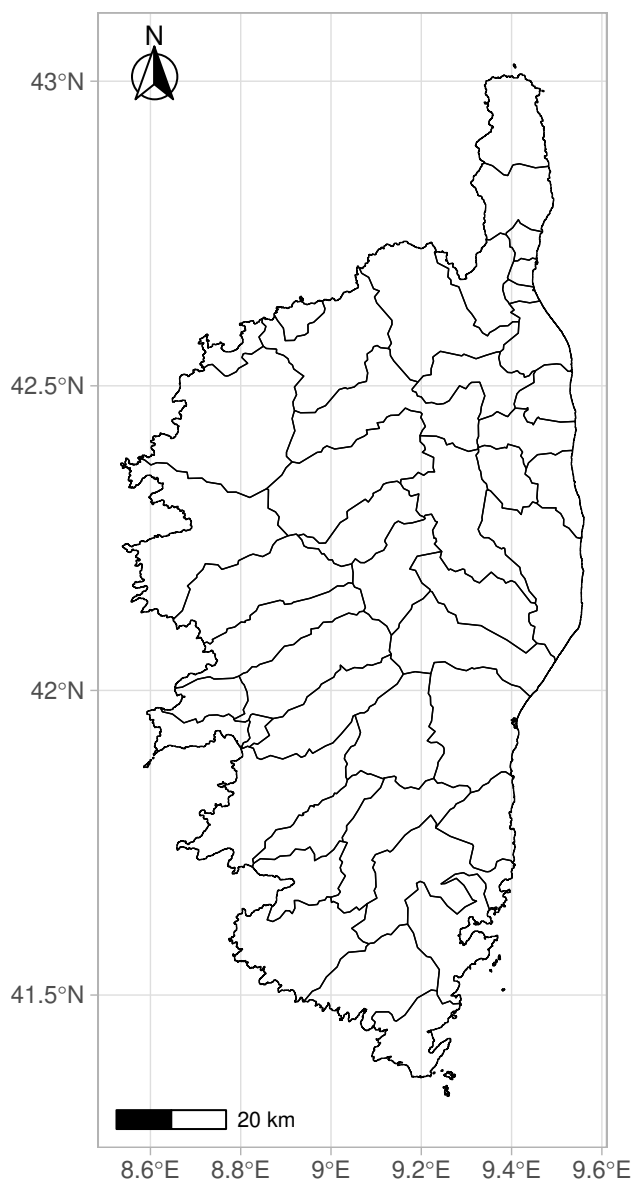


Figure 4: Corsica (Region of France) @ level 4

In order to extract some regions of a map we need to know them. The *listNames()* function allows this. The *subset* function is then used to extract the desired regions.

CAUTION: only the administrative levels that have been loaded in the *loadCountries* object can be listed. For instance, with a map loaded @ level 4, the level for *listNames* can be one of [0, 1, 2, 3, 4]. Names are given in the country's language or English.

```
listNames(Corsica, 2)
```

```
## [1] "Corse-du-Sud" "Haute-Corse"
```

```
HCorse <- gadm_subset(Corsica, regions="Haute-Corse", level=2)
```

```
gadm_plot(HCorse)
```

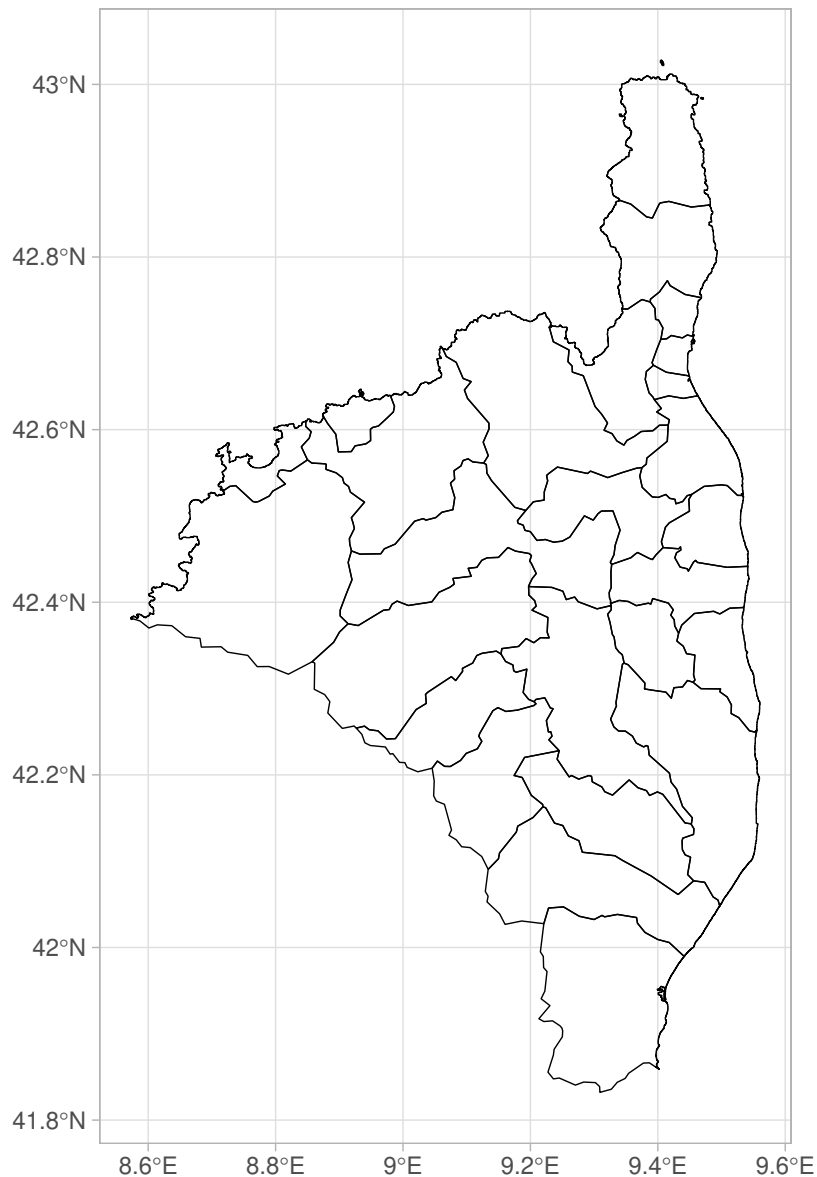


Figure 5: Corsica - Haute-Corse

Merging regions

```
UCorse <- gadm_union(Corsica, level=3, type="Arrondissements")  
gadm_plot(UCorse)
```

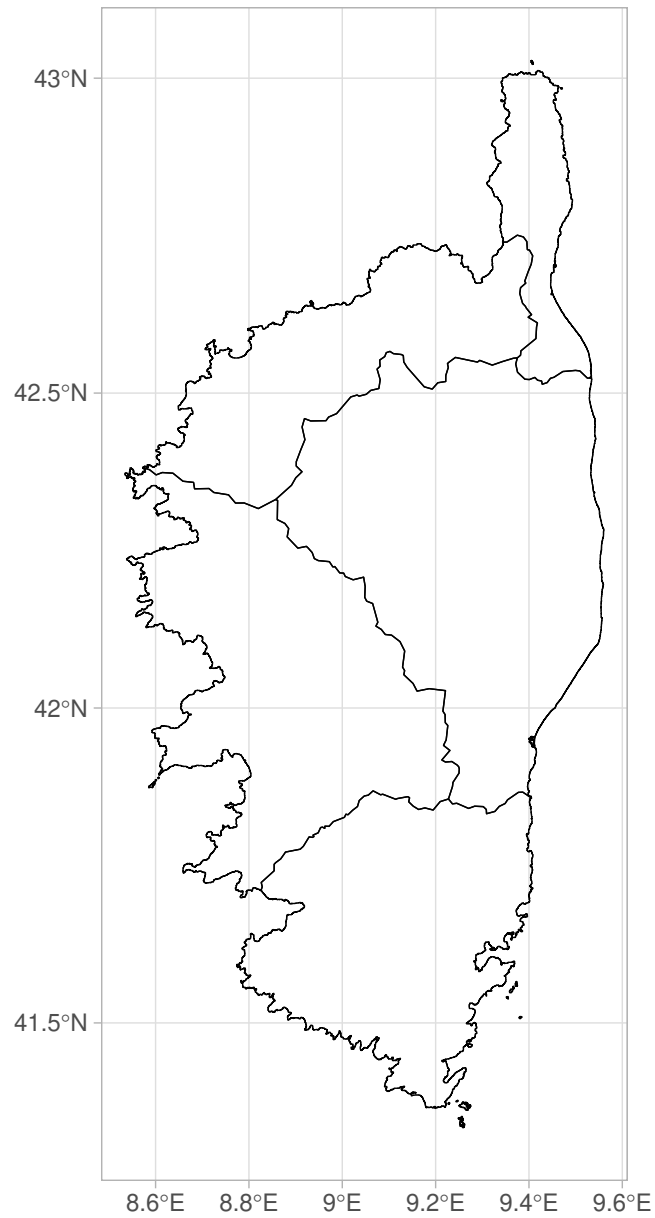


Figure 6: Corsica with districts only

Removing regions

```
listNames(Corsica, 3)

## [1] "Ajaccio" "Sartène" "Bastia" "Calvi" "Corte"

Corse_without_Corte <- gadm_remove(Corsica, regions="Corte", 3)
gadm_plot(Corse_without_Corte)
```

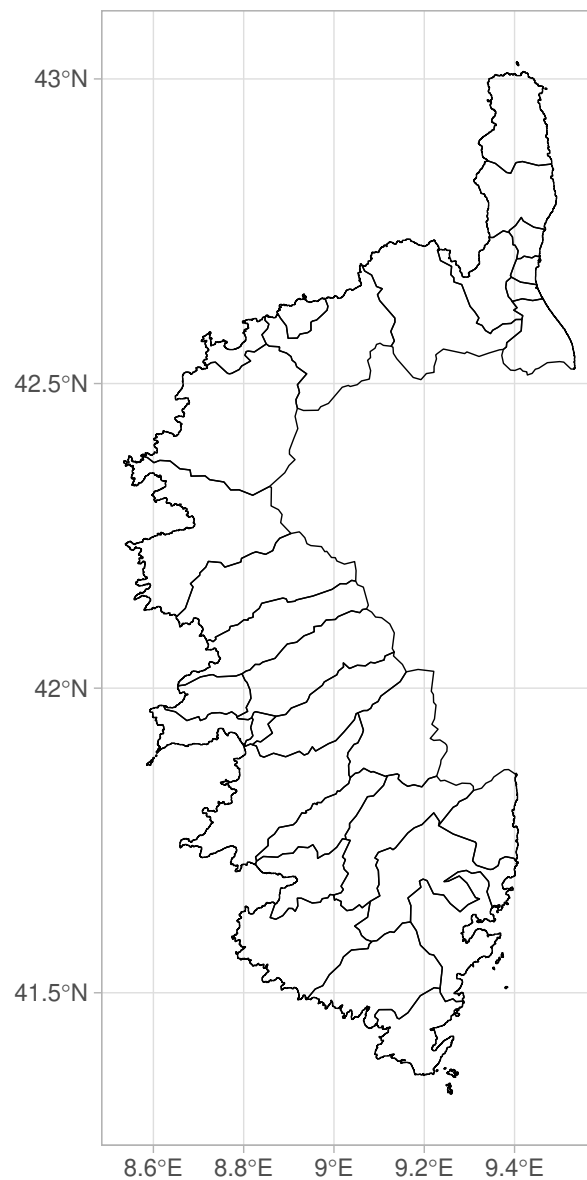


Figure 7: Corsica without district of Corte

Cropping an area

First get the bounding box of Corsica

```
gadm_getBbox(Corsica)
```

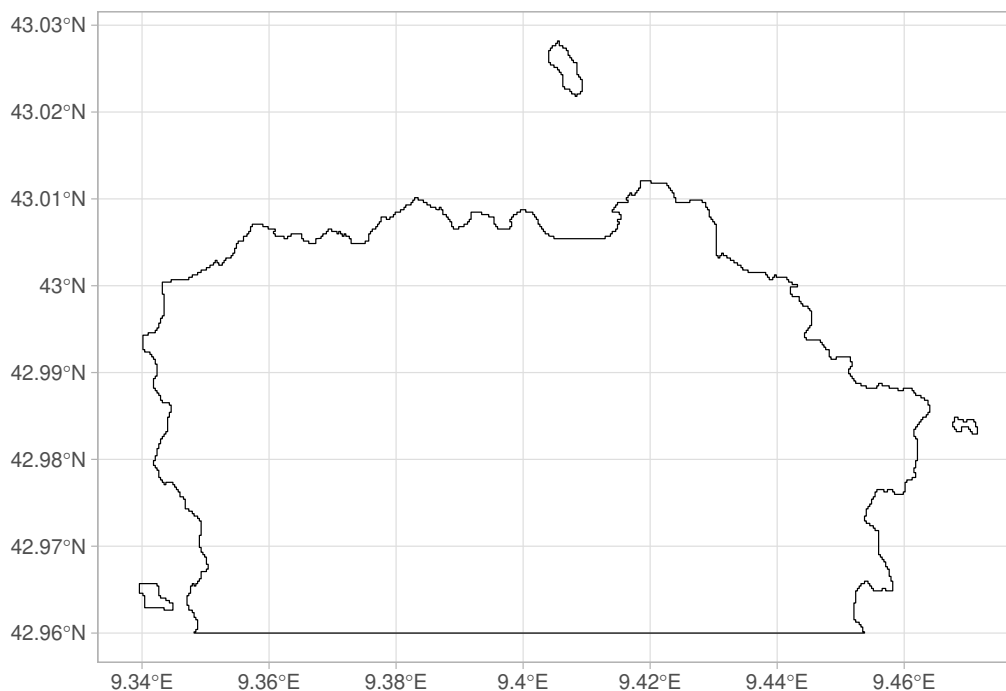
```
##      xmin      ymin      xmax      ymax  
## 8.534306 41.333752 9.560416 43.028194
```

And now, cropping at our custom coordinates

```
STUDY_AREA <- gadm_crop(Corsica, xmin=9.3, ymin=42.96, xmax=9.566, ymax=43.02819)
```

```
## although coordinates are longitude/latitude, st_intersection assumes that they are planar
```

```
gadm_plot(STUDY_AREA)
```



Converting longitudes to 0 - 360

```
library(GADMTTools)
FJI = gadm_sp_loadCountries("FJI", 1, basefile = "./")
gadm_plot(FJI, title = "Fidji Island with bad coordinates")
```



Figure 8: Fiji Islands, with polygons crossing the Date Line

```
FJI = gadm_longTo360(FJI)
gadm_plot(FJI, title = "Fidji Island with 0 - 360 coordinates")
```

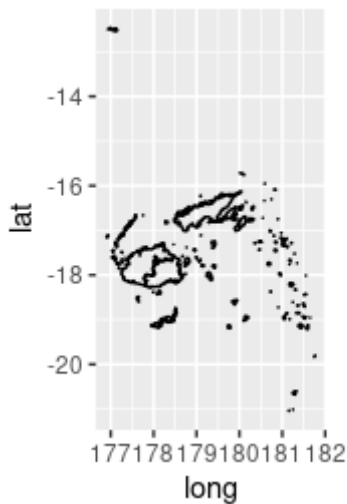


Figure 9: Fiji Islands, with polygons crossing the Date Line

Adding a background image from OpenstreetMap

```
library(GADMTools)
library(rosm)
FRA = gadm_sp_loadCountries("FRA", 2, basefile = "./")
BRE = gadm_subset(FRA, level=1, regions=c("Bretagne"))
BRE2 <- gadm_getBackground(BRE, "BRE", "osm")
gadm_plot(BRE2, title = "Map of Bretagne (FRANCE)")
```

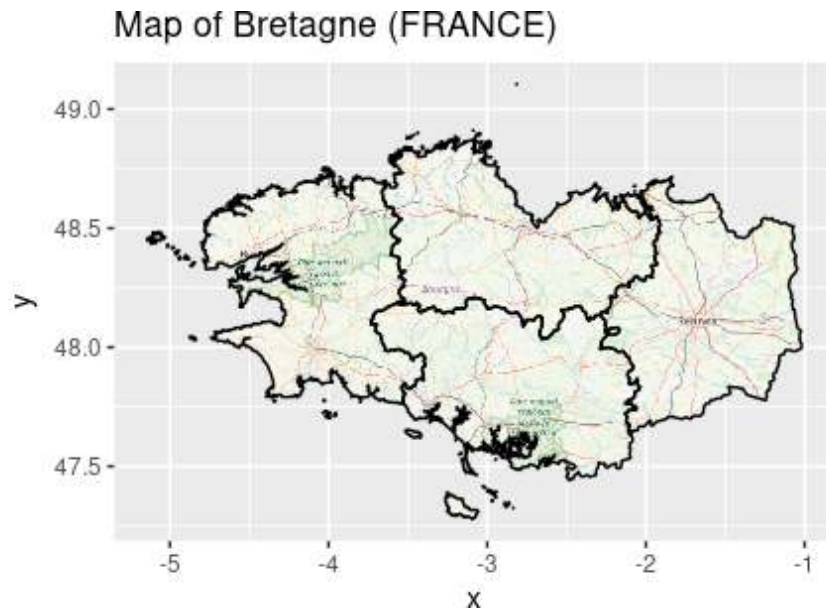


Figure 10: map of Bretagne with background from OSM @ level = 2

Remove a background previously loaded with `gadm_getBackground`

`gadm_removeBackground(x)`