More precisely *union* many spatial tables in R in a tidy way.

* dplyr::bind\_rows doesn’t work on sf objects ;
* base::rbind only work on two tables and so that’s not straightforward to use\*.

So we’ll use purrr::map and tidyr::unnest.

First get some data, the communes of three french départements :

library(tidyverse)

library(sf)

library(fs)

library(httr)

library(leaflet)

# https://fr.actualitix.com/blog/shapefiles-des-departements-de-france.html

url <- c("https://fr.actualitix.com/blog/actgeoshap/01-Ain.zip",

"https://fr.actualitix.com/blog/actgeoshap/73-savoie.zip",

"https://fr.actualitix.com/blog/actgeoshap/74-haute-savoie.zip")

dep <- str\_extract(url, "\\d{2}.\*$")

list(url, dep) %>%

pwalk(~ GET(.x, write\_disk(.y)))

walk(dep, unzip, junkpaths = TRUE, exdir = "shp")

We can then create a 3 rows data frame containing a list-column in which we store the sf object. Then we just unnest it. This operation erases the sf-class, we have to add it back.

res <- dir\_ls("shp", glob = "\*.shp") %>%

tibble(fname = .) %>%

mutate(data = map(fname, read\_sf)) %>%

unnest(data) %>%

st\_as\_sf() %>%

st\_set\_crs(2154)

write\_sf(res, "shp/3dep.shp")

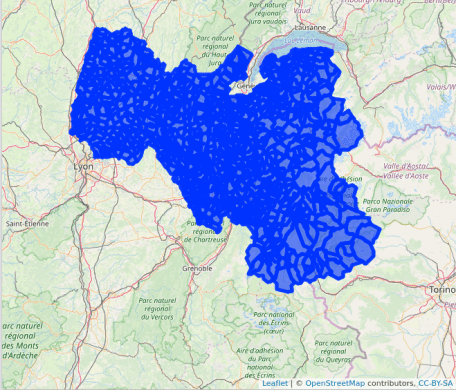
res %>%

st\_transform(4326) %>%

leaflet() %>%

addPolygons() %>%

addTiles()



Bonus : we have the source filename stored in the resulting shapefile.

\* We could have used

dir\_ls("shp", glob = "\*.shp") %>%

map(read\_sf) %>%

do.call(rbind, .)

but the column structure doesn’t match here…