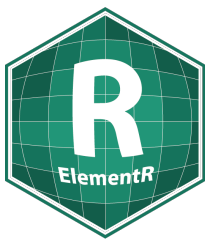
Création d'un packages R

ElementR



Timothée Giraud

Centre pour l'analyse spatiale et la géovisualisation

28 mai 2026

Objectif

Créer un package R permettant de reproduire ce type de carte.

Nedjib Sidi Moussa, 2022. Histoire algérienne de la France, PUF.

Materiel disponible

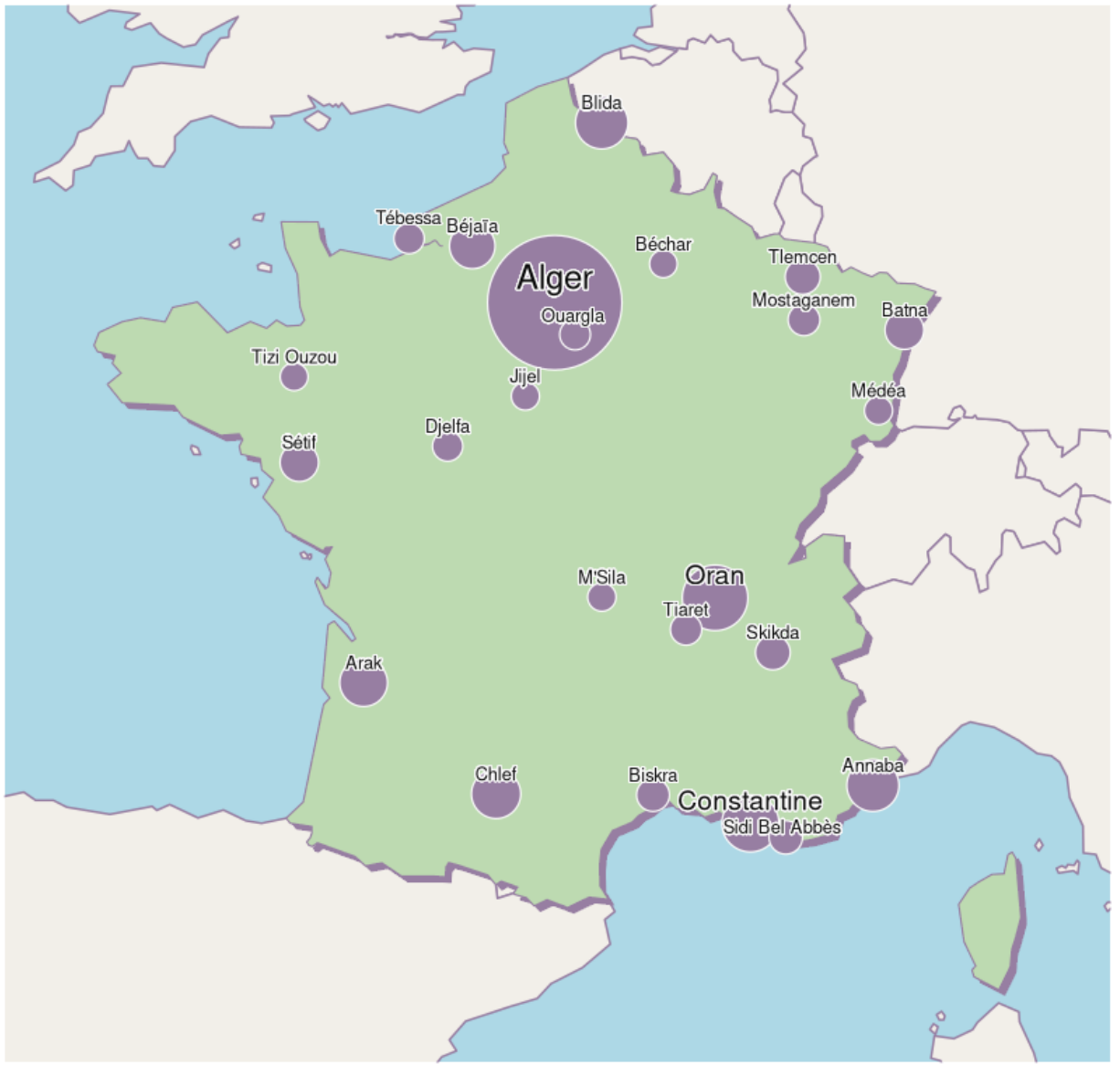
mapmix : un projet RStudio contenant des données, des fonctions et des scripts permettant de créer une carte

```
mapmix/
├── data
│   └── geocity.gpkg
├── mapmix.Rproj
├── my_functions.R
├── script1.R
└── script2.R
```

Les scripts

- Ouvrir le projet **mapmix** dans RStudio
- Ouvrir le fichier **script1.R**
- Exécuter le script

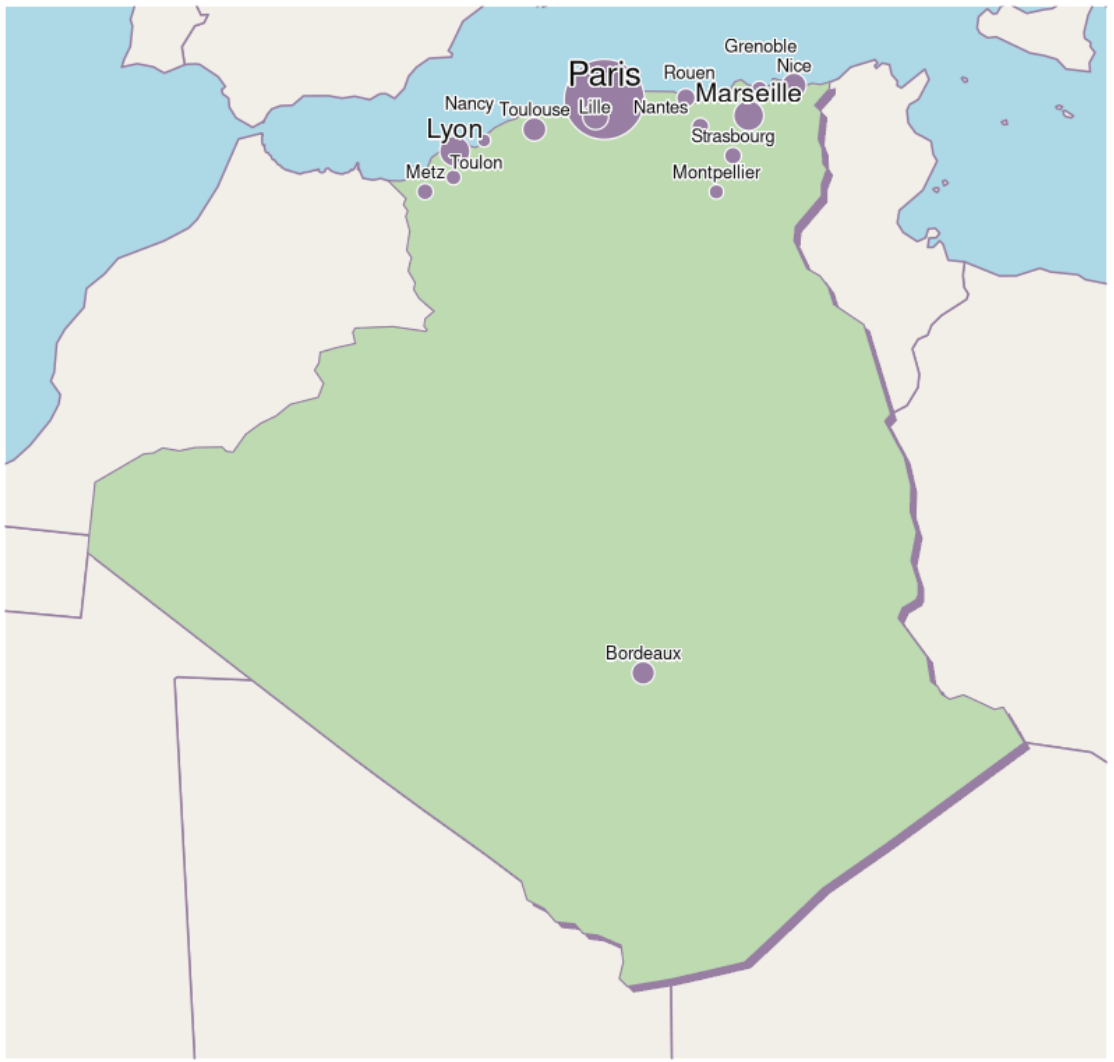
Les scripts



La carte produite par le script

Les scripts

- Modifier le script pour produire la carte suivante :



La carte produite en modifiant le script

Les fonctions

- Ouvrir les fichiers **script2.R** et **my_functions.R**
- Exécuter le fichier **script2.R**
- Quelles modifications ont été apportées au script initial?
- Quels sont les packages nécessaires ?
- Comment savoir ce que sont x, y et n ?

Le package

Il est préférable (voire nécessaire) d'avoir les dernières versions de R, des packages et de RStudio pour développer un package.

Nous avons besoin des packages `devtools` et `usethis` qui faciliteront plusieurs étapes du processus de développement.

- Créer le squelette du package `citymix` avec

```
usethis::create_package("/chemin/du/projet_de_package/citymix")
```

- Observer la structure du package

Description et informations sur le package

- Ouvrir le fichier **DESCRIPTION** et l'éditer avec les bonnes informations?

```
Package: citymix
Title: Plot a Map on a Map
Version: 0.1.0
Authors@R:
    person(given = "Timothée",
             family = "Giraud",
             role = c("aut", "cre"),
             email = "timothee.giraud@cnrs.fr",
             comment = c(ORCID = "0000-0002-1932-3323"))
Description: Create a map of a country with cities from another country,
    respecting the urban hierarchy.
License: GPL (>= 3)
Encoding: UTF-8
Roxygen: list(markdown = TRUE)
RoxygenNote: 8.0.0
```

Les données et les fonctions du package

- Copier le fichier **data/geocity.gpkg** du projet **mapmix** dans un nouveau dossier **/inst/gpkg/** dans le projet du package.
- Copier le fichier **my_functions.R** du projet **mapmix** dans le dossier **/R** du projet du package.

```
citymix
├── citymix.Rproj
├── DESCRIPTION
├── .gitignore
├── inst
│   └── gpkg
│       └── geocity.gpkg
├── NAMESPACE
├── R
│   └── my_functions.R
└── .Rbuildignore
```

Adapter les chemins vers les données

- Ouvrir le fichier **R/my_functions.R**
- Changer les lignes suivantes

```
countries <- st_read("data/geocity.gpkg", "countries", quiet = TRUE)
cities <-  st_read("data/geocity.gpkg", "cities", quiet = TRUE)
```

par celles-ci

```
countries <- st_read(system.file("gpkg/geocity.gpkg", package = "citymix"),
                     "countries", quiet = TRUE)
cities <-  st_read(system.file("gpkg/geocity.gpkg", package = "citymix"),
                   "cities", quiet = TRUE)
```

Les fonctions des autres packages

- Préfixer toutes les fonctions venant du package `sf` par `sf::`

```
focus <- st_transform(x = focus, prj)
country <- st_transform(x = country, prj)
```

devient

```
focus <- sf::st_transform(x = focus, prj)
country <- sf::st_transform(x = country, prj)
```

- Retirer l'appel à la bibliothèque `sf` du fichier **my_functions.R**
  ~~`library(sf)`~~
- Ajouter la ligne suivante au fichier **DESCRIPTION**

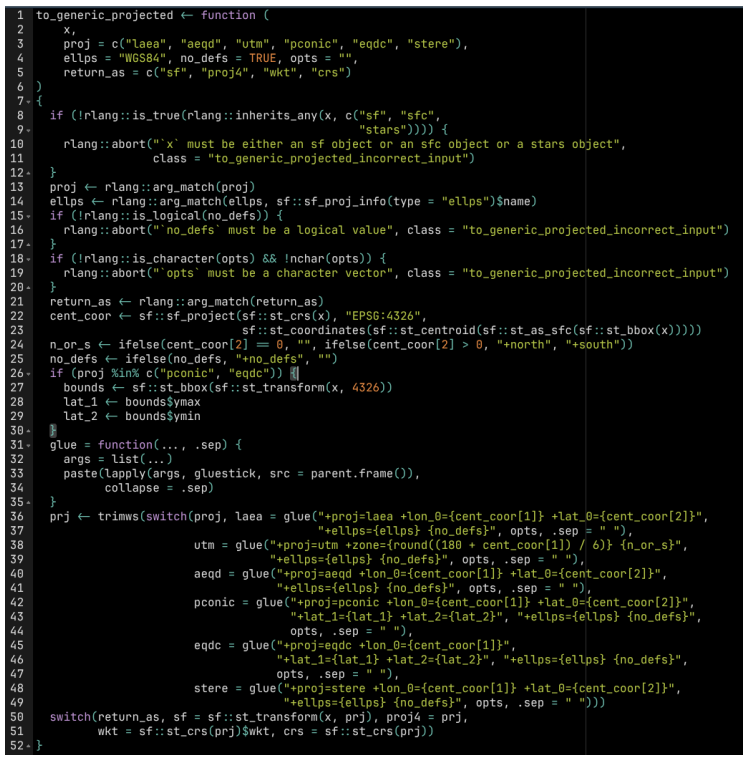
```
Imports: sf
```

Les fonctions des autres packages

- Inspecter le code de la fonction interne `tmap:::to_generic_projected()`

Les fonctions des autres packages

Pour nos besoins, on peut de passer de :



```
to_generic_projected <- function (
    x,
    proj = c("laea", "aeqd", "utm", "pconic", "eqdc", "stere"),
    ellps = "WGS84", no_defs = TRUE, opts = "",
    return_as = c("sf", "proj4", "wkt", "crs")
)
{
  if (!rlang::is_true(rlang::inherits_any(x, c("sf", "sfc",
                                               "stars")))) {
    rlang::abort("`x` must be either an sf object or an sfc object or a stars object",
                 class = "to_generic_projected_incorrect_input")
  }
  proj <- rlang::arg_match(proj)
  ellps <- rlang::arg_match(ellps, sf::sf_proj_info(type = "ellps")$name)
  if (!rlang::is_logical(no_defs)) {
    rlang::abort("`no_defs` must be a logical value", class = "to_generic_projected_incorrect_input")
  }
  if (!rlang::is_character(opts) && !nchar(opts)) {
    rlang::abort("`opts` must be a character vector", class = "to_generic_projected_incorrect_input")
  }
  return_as <- rlang::arg_match(return_as)
  cent_coor <- sf::sf_project(sf::st_crs(x), "EPSG:4326",
                              sf::st_coordinates(sf::st_centroid(sf::st_as_sfc(sf::st_bbox(x)))))
  n_or_s <- ifelse(cent_coor[2] == 0, "", ifelse(cent_coor[2] > 0, "+north", "+south"))
  no_defs <- ifelse(no_defs, "+no_defs", "")
  if (proj %in% c("pconic", "eqdc")) {
    bounds <- sf::st_bbox(sf::st_transform(x, 4326))
    lat_1 <- bounds$ymax
    lat_2 <- bounds$ymin
  }
  glue = function(..., .sep) {
    args = list(...)
    paste(lapply(args, gluestick, src = parent.frame()),
          collapse = .sep)
  }
  prj <- trimws(switch(proj, laea = glue("+proj=laea +lon_0={cent_coor[1]} +lat_0={cent_coor[2]}",
                                         "+ellps={ellps} {no_defs}", opts, .sep = " "),
                       utm = glue("+proj=utm +zone={round((180 + cent_coor[1]) / 6)} {n_or_s}",
                                  "+ellps={ellps} {no_defs}", opts, .sep = " "),
                       aeqd = glue("+proj=aeqd +lon_0={cent_coor[1]} +lat_0={cent_coor[2]}",
                                   "+ellps={ellps} {no_defs}", opts, .sep = " "),
                       pconic = glue("+proj=pconic +lon_0={cent_coor[1]} +lat_0={cent_coor[2]}",
                                     "+lat_1={lat_1} +lat_2={lat_2}", "+ellps={ellps} {no_defs}",
                                     opts, .sep = " "),
                       eqdc = glue("+proj=eqdc +lon_0={cent_coor[1]}",
                                   "+lat_1={lat_1} +lat_2={lat_2}", "+ellps={ellps} {no_defs}",
                                   opts, .sep = " "),
                       stere = glue("+proj=stere +lon_0={cent_coor[1]} +lat_0={cent_coor[2]}",
                                    "+ellps={ellps} {no_defs}", opts, .sep = " ")))
  switch(return_as, sf = sf::st_transform(x, prj), proj4 = prj,
         wkt = sf::st_crs(prj)$wkt, crs = sf::st_crs(prj))
}
```

à :

```
cent_coor <- sf::st_coordinates(sf::st_centroid(sf::st_as_sfc(sf::st_bbox(x))))
prj <- paste0("+proj=laea +lon_0=", cent_coor[1], " +lat_0=", cent_coor[2],
              " +ellps=WGS84 +no_defs")
prj <- sf::st_crs(prj)$wkt
```

Les fonctions des autres packages

- Remplacez

```
prj <- tmap:::to_generic_projected(focus, return_as = "wkt")
```

par

```
cent_coor <- sf::st_coordinates(sf::st_centroid(sf::st_as_sfc(sf::st_bbox(focus))))
prj <- paste0("+proj=laea +lon_0=", cent_coor[1], " +lat_0=", cent_coor[2],
              " +ellps=WGS84 +no_defs")
prj <- sf::st_crs(prj)$wkt
```

Documenter la fonction

- Cliquer n’importe où dans la fonction
- [Code] => [Insert Roxygen skeleton]

```
#' Title
#'
#' @param x
#' @param y
#' @param n
#'
#' @returns
#' @export
#'
#' @examples
```

Documenter la fonction

```
#' Create a map of country with cities from another country
#'
#' Create a map of a country with cities from another country,
#' respecting the urban hierarchy.
#'
#' @param x ISO 3166-1 alpha-3 code of the country to map
#' @param y ISO 3166-1 alpha-3 code of the country used for cities names and
#' populations
#' @param n maximum number of cities to plot
#'
#' @returns Nothing is returned, a map is plotted
#' @export
#'
#' @examples
#' citymix("FRA", "DZA", n = 5)
```

Les valeurs par défaut

- Mettre une valeur par défaut pour l'argument `n` :

```
citymix <- function(x, y, n = 5) {
  ...
}
```

Les fonctions des autres packages

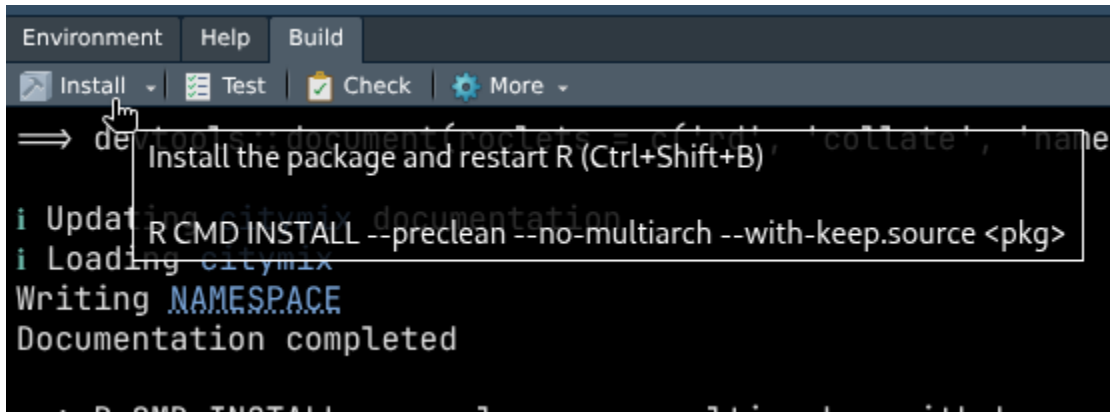
- Indiquer les fonctions de `mapsf` utilisées par la fonction `citymix()` en rajoutant la ligne suivante dans la documentation:

```
#' @importFrom mapsf mf_map mf_shadow mf_label mf_theme
```

- Retirer l'appel à la bibliothèque `mapsf` du fichier **my_functions.R**
  ~~`library(mapsf)`~~
- Modifier la ligne des imports du fichier **DESCRIPTION** pour rajouter `mapsf`.

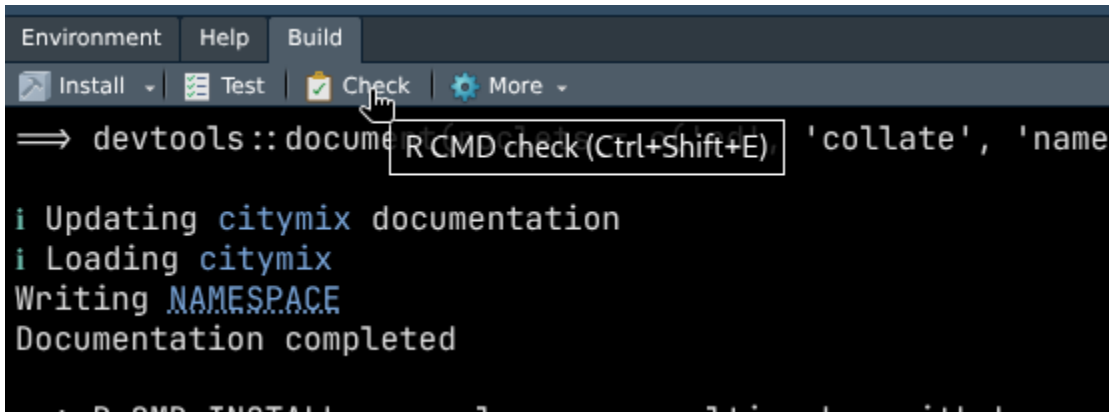
```
Imports: sf, mapsf
```

Contruire et installer le package




```
citymix
├── citymix.Rproj
├── DESCRIPTION
├── .gitignore
├── inst
│   └── gpkg
│       └── geocity.gpkg
├── man
│   └── citymix.Rd
├── NAMESPACE
├── R
│   └── my_functions.R
└── .Rbuildignore
```

Vérifier le package



Et ensuite ?

- Mettre le package sur une forge
- Construire un README qui présente le package
- Contruire des tests
- Utiliser l'intégration continue pour tester le package quand on modifie son code
- Construire un site web de démonstration du package
- Utiliser le déploiement continue pour mettre à jour le package site web quand on modifie le code du package
- Déployer le site sur r-universe
- Soumettre le package au CRAN