R pour le Géospatial 🌐

Import

Etienne Racine

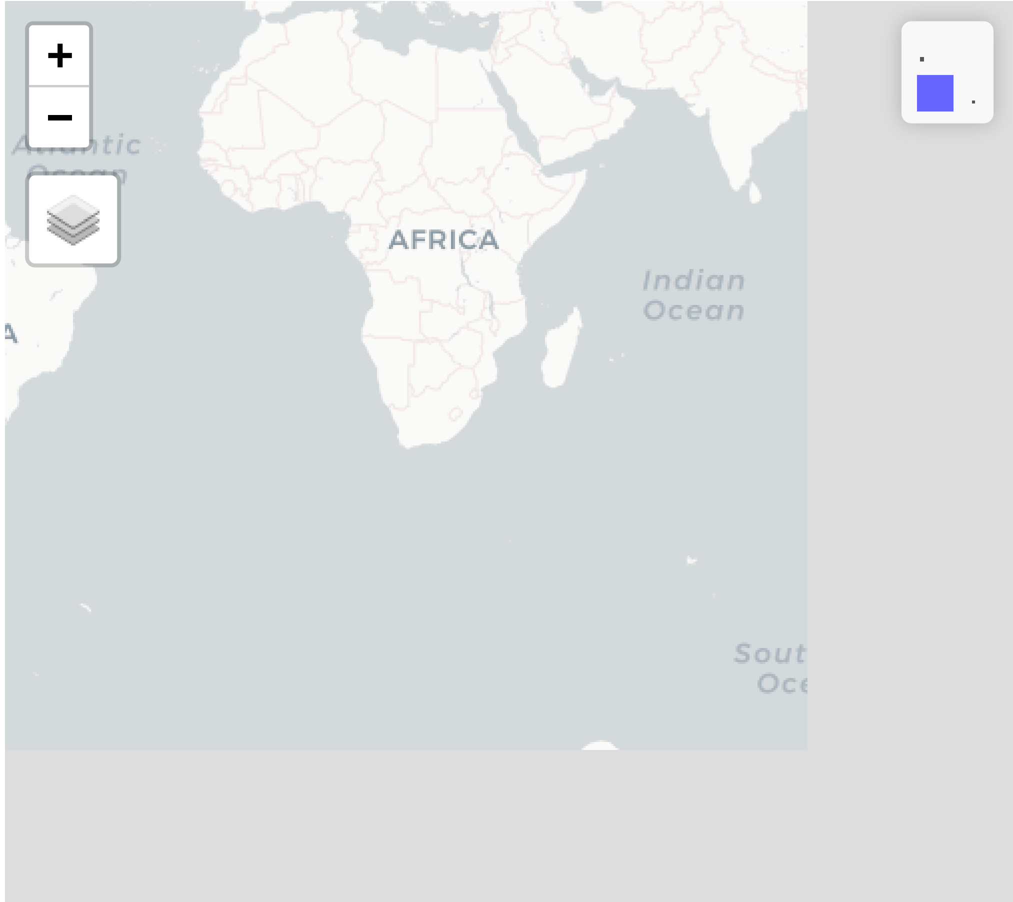
Shapefile

```
path <- system.file("shape", "storms_xyz.shp", package = "sf")
shp <- read_sf(path)
print(shp)
```

```
## Simple feature collection with 71 features and 0 fields
## geometry type:  LINESTRING
## dimension:      XYZ
## bbox:           xmin: -102.2 ymin: 8.3 xmax: 0 ymax: 59.5
## epsg (SRID):    NA
## proj4string:    NA
## First 10 features:
##                           geometry
## 1  LINESTRING Z (-50.8 20.1 10...
## 2  LINESTRING Z (-77.4 14.3 10...
## 3  LINESTRING Z (-62.7 14.7 10...
## 4  LINESTRING Z (-72.5 25.5 10...
## 5  LINESTRING Z (-38 12.4 1008...
## 6  LINESTRING Z (-38 15.5 1008...
## 7  LINESTRING Z (-36.7 28.9 10...
## 8  LINESTRING Z (-27.4 12.9 10...
## 9  LINESTRING Z (-43.4 23.9 10...
## 10 LINESTRING Z (-31.7 10.7 10...
```

```r
shp %>%
  st_set_crs(4326) %>%
  mapview()
```

```
## Warning in cbind(`Feature ID` = fid, mat): number of rows of result is not
## a multiple of vector length (arg 1)
```

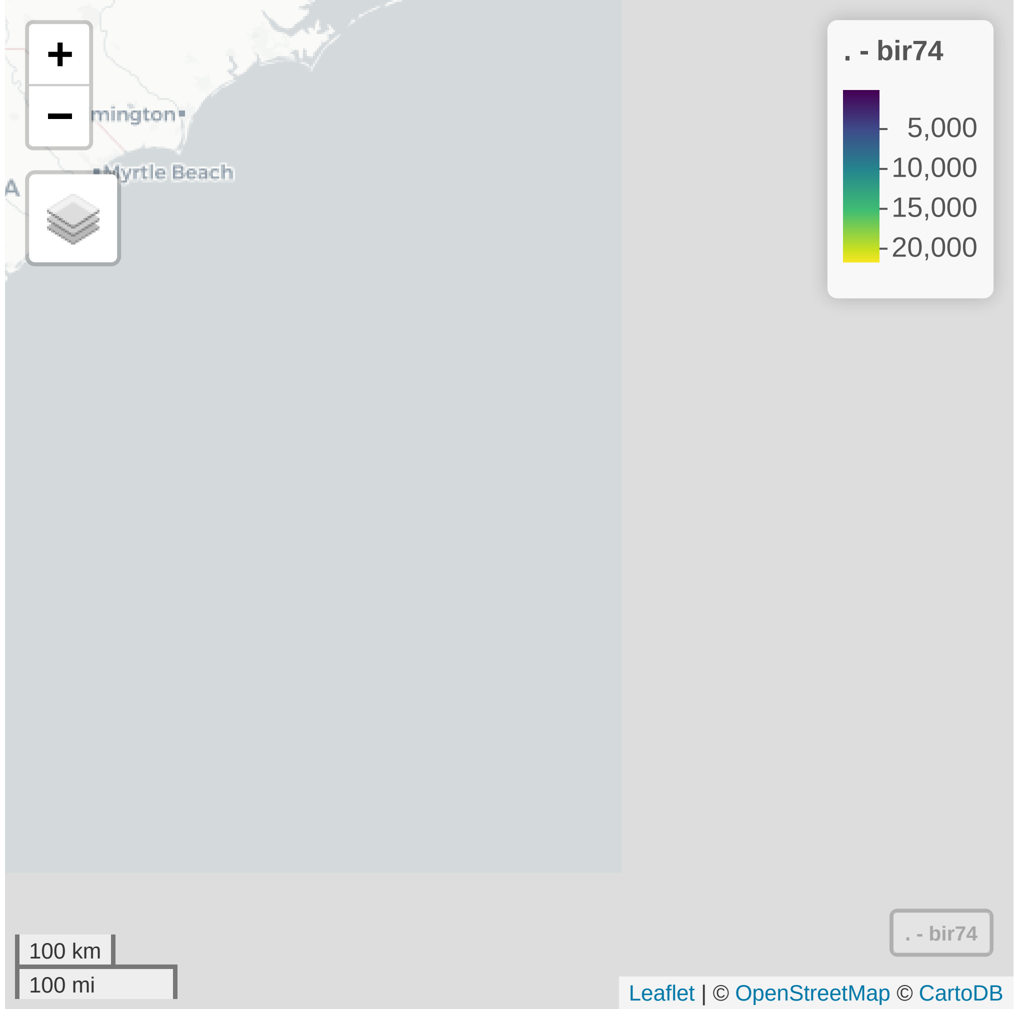


GPKG

```
path <- system.file("gpkg", "nc.gpkg", package = "sf")
gpkg <- read_sf(path) %>% #<<
  set_names(tolower)
```

```
## Simple feature collection with 100 features and 14 fields
## geometry type:  MULTIPOLYGON
## dimension:      XY
## bbox:           xmin: -84.32385 ymin: 33.88199 xmax: -75.45698 ymax: 36.58965
## epsg (SRID):    4267
## proj4string:    +proj=longlat +datum=NAD27 +no_defs
## # A tibble: 100 x 15
##     area perimeter cnty_ cnty_id name  fips  fipsno cress_id bir74 sid74
##    <dbl>     <dbl> <dbl>   <dbl> <chr> <chr>  <dbl>    <int> <dbl> <dbl>
##  1 0.114      1.44  1825    1825 Ashe  37009  37009        5  1091     1
##  2 0.061      1.23  1827    1827 Alle… 37005  37005        3   487     0
##  3 0.143      1.63  1828    1828 Surry 37171  37171       86  3188     5
##  4 0.07       2.97  1831    1831 Curr… 37053  37053       27   508     1
##  5 0.153      2.21  1832    1832 Nort… 37131  37131       66  1421     9
##  6 0.097      1.67  1833    1833 Hert… 37091  37091       46  1452     7
##  7 0.062      1.55  1834    1834 Camd… 37029  37029       15   286     0
##  8 0.091      1.28  1835    1835 Gates 37073  37073       37   420     0
##  9 0.118      1.42  1836    1836 Warr… 37185  37185       93   968     4
## 10 0.124      1.43  1837    1837 Stok… 37169  37169       85  1612     1
```

```r
gpkg %>%
  mapview(zcol = "bir74")
```



Database

Si vous pouvez installer docker sur votre système, harbor permet de contrôler un conteneur postgis à partir de R.

```r
# install_github("wch/harbor", force_deps = TRUE)

library(harbor)
library(DBI)

dock <- docker_run(
  image = "kartoza/postgis",
  docker_opts = c(
    "-p", "5432:5432",
    "-e", "POSTGRES_DBNAME=postgis",
    "-e", "POSTGRES_USER=postgres"),
  detach = TRUE
)
# wait for the database to run
Sys.sleep(10)
```

Database

```r
pg <- dbConnect(
  RPostgres::Postgres(),
  host = "localhost",
  dbname = "postgis",
  user = "postgres",
  password = "docker")
```

Database - I/O

```r
# remember `set_names(tolower)`
st_write(gpkg, pg, "nc")
```

```
## Note: method with signature 'DBIObject#sf' chosen for function 'dbDataType',
##  target signature 'PqConnection#sf'.
##  "PqConnection#ANY" would also be valid
```

```r
st_read(pg, "nc")
```

```
## Simple feature collection with 100 features and 14 fields
## geometry type:  MULTIPOLYGON
## dimension:      XY
## bbox:           xmin: -84.32385 ymin: 33.88199 xmax: -75.45698 ymax: 36.58965
## epsg (SRID):    4267
## proj4string:    +proj=longlat +datum=NAD27 +no_defs
## First 10 features:
##     area perimeter cnty_ cnty_id        name  fips fipsno cress_id bir74
## 1  0.114     1.442  1825    1825        Ashe 37009  37009        5  1091
## 2  0.061     1.231  1827    1827   Alleghany 37005  37005        3   487
## 3  0.143     1.630  1828    1828       Surry 37171  37171       86  3188
## 4  0.070     2.968  1831    1831   Currituck 37053  37053       27   508
```

Database

```
# remember `set_names(tolower)`
q <-
  "select name,
    st_area(geom::geography) as area,
    st_buffer(geom::geography, 3000)::geometry
  from nc"
buf <- st_read(pg, query = q)
```

Push R code to database

`dplyr` fonctionne à la fois sur une table locale ou distante.

```
nc <- tbl(pg, "nc")

nc
```

```
## # Source:   table<nc> [?? x 15]
## # Database: postgres [postgres@localhost:5432/postgis]
##     area perimeter cnty_ cnty_id name  fips  fipsno cress_id bir74 sid74
##    <dbl>     <dbl> <dbl>   <dbl> <chr> <chr>  <dbl>    <int> <dbl> <dbl>
##  1 0.114      1.44  1825    1825 Ashe  37009  37009        5  1091     1
##  2 0.061      1.23  1827    1827 Alle… 37005  37005        3   487     0
##  3 0.143      1.63  1828    1828 Surry 37171  37171       86  3188     5
##  4 0.07       2.97  1831    1831 Curr… 37053  37053       27   508     1
##  5 0.153      2.21  1832    1832 Nort… 37131  37131       66  1421     9
##  6 0.097      1.67  1833    1833 Hert… 37091  37091       46  1452     7
##  7 0.062      1.55  1834    1834 Camd… 37029  37029       15   286     0
##  8 0.091      1.28  1835    1835 Gates 37073  37073       37   420     0
##  9 0.118      1.42  1836    1836 Warr… 37185  37185       93   968     4
## 10 0.124      1.43  1837    1837 Stok… 37169  37169       85  1612     1
## # … with more rows, and 5 more variables: nwbir74 <dbl>, bir79 <dbl>,
## #   sid79 <dbl>, nwbir79 <dbl>, geom <chr>
```

```r
buf <- nc %>%
  filter(area > 0.15) %>%
  mutate(large = geom %>% st_transform(32119L) %>% st_buffer(3000) %>% st_transform(4326L))

buf %>% show_query()
```

```
## <SQL>
## SELECT "area", "perimeter", "cnty_", "cnty_id", "name", "fips", "fipsno", "cress_id", "bir74"
## FROM "nc"
## WHERE ("area" > 0.15)
```

```
## # Source:   lazy query [?? x 16]
## # Database: postgres [postgres@localhost:5432/postgis]
##     area perimeter cnty_ cnty_id name  fips  fipsno cress_id bir74 sid74
##    <dbl>     <dbl> <dbl>   <dbl> <chr> <chr>  <dbl>    <int> <dbl> <dbl>
##  1 0.153      2.21  1832    1832 Nort… 37131  37131       66  1421     9
##  2 0.153      1.62  1839    1839 Rock… 37157  37157       79  4449    16
##  3 0.19       2.20  1846    1846 Hali… 37083  37083       42  3608    18
##  4 0.201      1.80  1968    1968 Rand… 37151  37151       76  4456     7
##  5 0.199      1.98  1874    1874 Wilk… 37193  37193       97  3146     4
##  6 0.17       1.68  1903    1903 Guil… 37081  37081       41 16184    23
##  7 0.18       2.15  1905    1905 Bert… 37015  37015        8  1324     6
##  8 0.219      2.13  1938    1938 Wake  37183  37183       92 14484    16
##  9 0.155      1.78  1947    1947 Ired… 37097  37097       49  4139     4
## 10 0.18       2.14  1973    1973 Chat… 37037  37037       19  1646     2
```

Collect

Rapatrier les données dans R (en mémoire).

```r
buf <- buf %>%
  collect() %>%
  mutate(large = st_as_sfc(sf:::as_wkb(large), EWKB = TRUE)) %>%
  st_as_sf()
```

Fermer le conteneur

Une fois qu'on a terminé du conteneur

```
dbDisconnect(pg)
container_stop(dock)
```

À vous

- Les données `storms` proviennent du United States National Oceanic and Atmospheric Administration (NOAA) et sont disponibles dans `dplyr` dans un format de tables. Pouvez-vous créer une carte qui affiche ces données? indice: `dplyr::storms`.
- Essayez vos propres données ou les données disponibles dans le portail des données ouvertes. Par exemple: www.donneesquebec.ca