Package SF

Kabirou Kanlanfeyi, Jordy Hounsinou

20/12/2020

I- Présentation du package SF

Utilisé pour améliorer les fonctionnalités fournies par le package SP, le package SF permet de gagner du temps en proposant plusieurs fonctions qui permettent de gagner du temps. Il intègre des fonctions qui permettent de faire non seulement de visualisation de données géographiques couplé avec d'autres librairies mais aussi des opérations géométriques, les mesures, la création d'objets mais également des fonctions de confirmation.

Les principales fonctionnalités sont les suivantes:

- représentation des entités simples sous forme d'enregistrements dans un data.frame ou tibble avec une liste-colonne de géométrie.
- représentation native dans R des 17 types d'entités simples pour toutes les dimensions (XY, XYZ, XYM, XYZM)
- servir d'interface avec GEOS pour prendre en charge les opérations géométriques, y compris le DE9-IM
- servir d'interface vers GDAL, prenant en charge toutes les options de pilote, Date et POSIXct et colonnes de liste
- servir d'interface avec PRØJ pour les conversions et transformations du système de référence de coordonnées
- utiliser des sérialisations binaires bien connues écrites en C ++ / Rcpp pour des E / S rapides avec GDAL et GEOS
- lire et écrire dans des bases de données spatiales telles que PostGIS à l'aide de DBI

II- Illustration du package *SF*

Données utilisées

Nous allons utiliser dans nos exemples, deux principales sources de données:
- La carte de l'IDF (format shp) - La carte de Paris qui contient la liste des différents arrondissements (format shp) - Les données sur les musées en île de France

Pour des opérations, nous allons diviser les arrondissements de paris en 2 parties. une contenant 7 arronssements et l'autre contenant les 13 autres restants.

Nous appliquerons sur elles les principales fonctions inclues dans le package *SF*

Importation des données

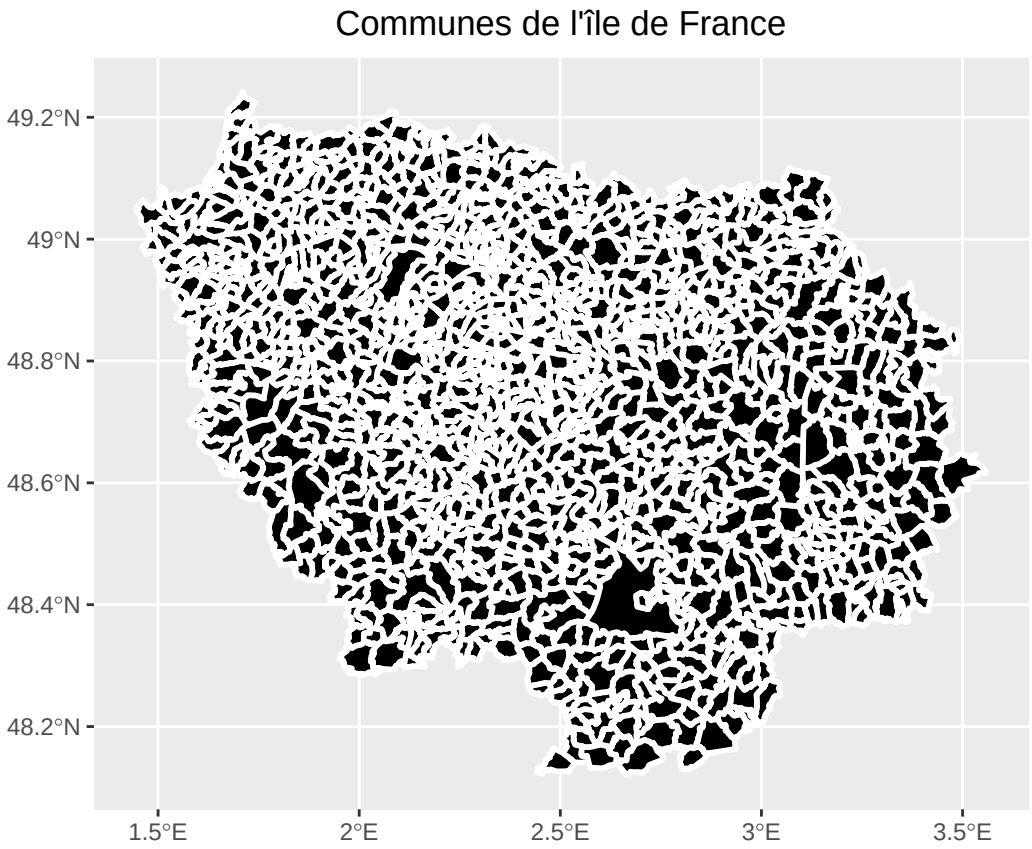
```
library(sf)
```

```
## Linking to GEOS 3.8.1, GDAL 3.1.4, PROJ 6.3.1
```

```
library(ggplot2)

#Communes de l'île de france
IDF <- st_read("Communes_IDF.shp")
```

```
## Reading layer 'Communes_IDF' from data source '/Users/kabkabirou/Documents/ProjetsR/SP and SF/Commun
## Simple feature collection with 1304 features and 36 fields
## geometry type:  POLYGON
## dimension:      XY
## bbox:           xmin: 1.446358 ymin: 48.1203 xmax: 3.559018 ymax: 49.24165
## geographic CRS: WGS 84
```

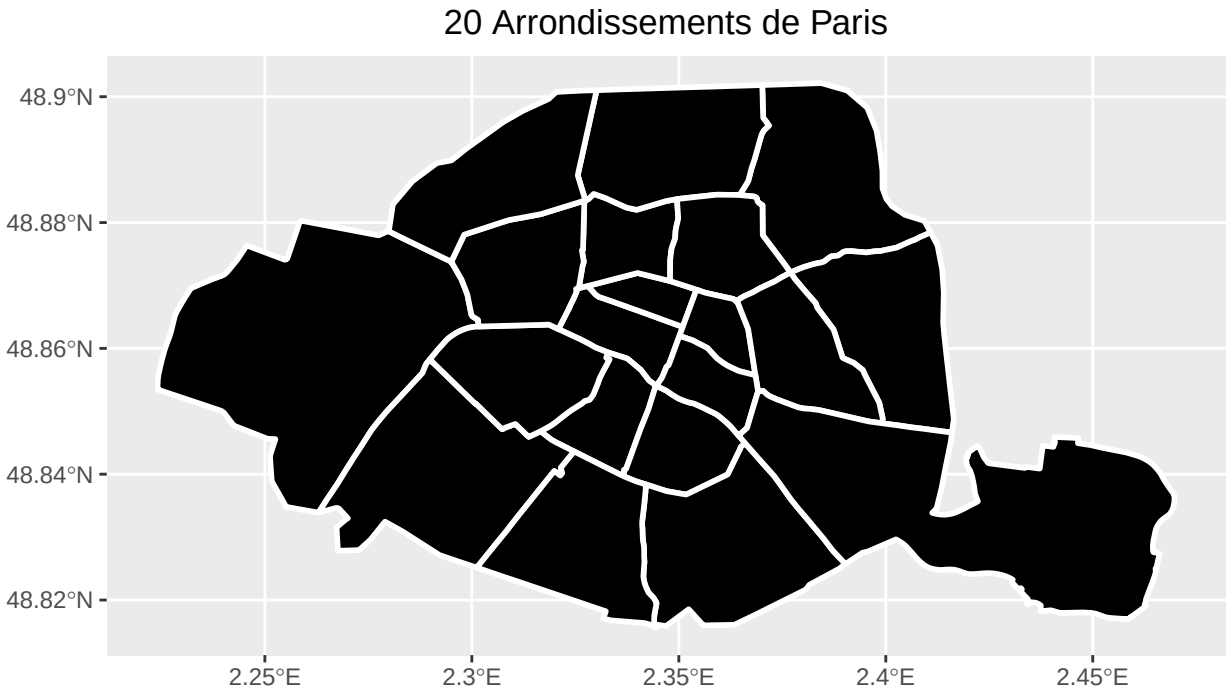
```
ggplot() +
  geom_sf(data = IDF, size = 1, color = "white", fill = "black") +
  ggtitle("Communes de l'île de France") +
  #Centrer le Titre de la carte
  theme(plot.title = element_text(hjust = 0.5)) +
  coord_sf()
```




```
#Arrondissements de Paris
AR <- st_read("arrondissements/arrondissements.shp")
```

```
## Reading layer ‘arrondissements’ from data source ‘/Users/kabkabirou/Documents/ProjetsR/SP and SF/arr
## Simple feature collection with 20 features and 8 fields
## geometry type:  POLYGON
## dimension:      XY
## bbox:           xmin: 2.224078 ymin: 48.81558 xmax: 2.469761 ymax: 48.90216
## geographic CRS: WGS 84
```

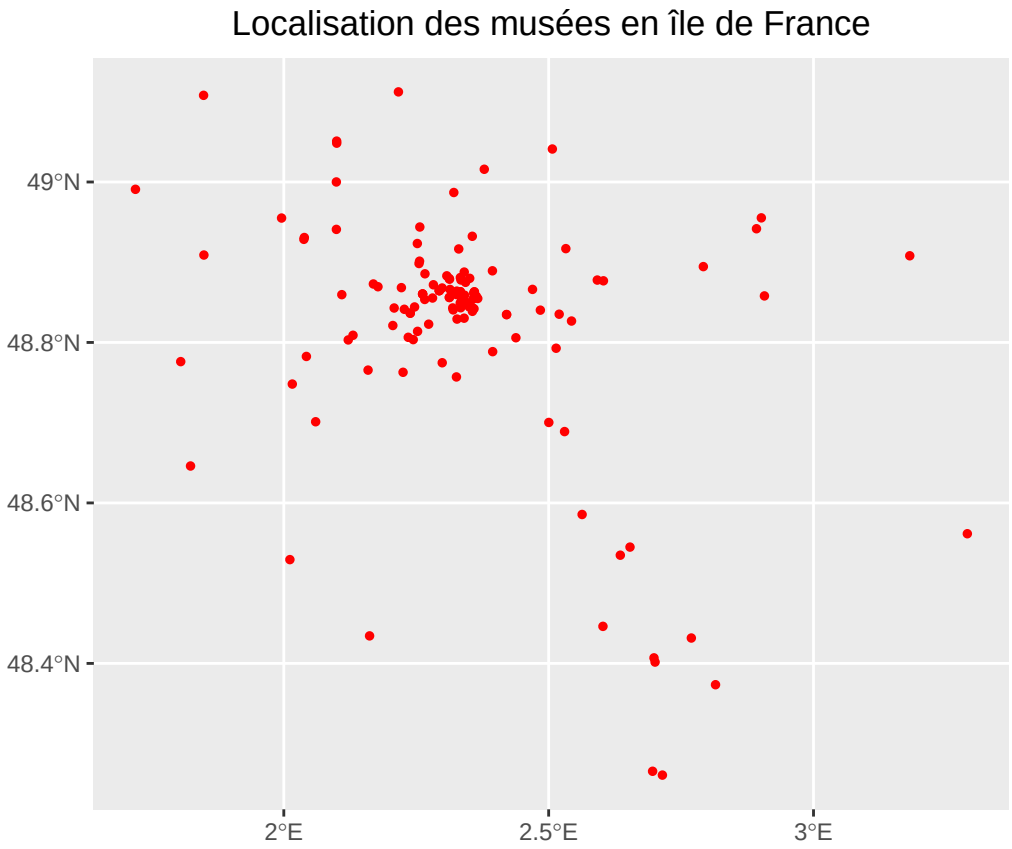
```
ggplot() +
  geom_sf(data = AR, size = 1, color = "white", fill = "black") +
  theme(plot.title = element_text(hjust = 0.5)) +
  ggtitle("20 Arrondissements de Paris") +
  coord_sf()
```




```
#Musées d'île de France
musees <- st_read("musees/liste-et-localisation-des-musees-de-france.shp")
```

```
## Reading layer 'liste-et-localisation-des-musees-de-france' from data source '/Users/kabkabirou/Docum
## replacing null geometries with empty geometries
## Simple feature collection with 143 features and 15 fields (with 15 geometries empty)
## geometry type:  POINT
## dimension:      XY
## bbox:           xmin: 1.719901 ymin: 48.26079 xmax: 3.290374 ymax: 49.11228
## geographic CRS: WGS 84
```

```
ggplot() +
  geom_sf(data = musees, size = 1, color = "red") +
  ggtitle("Localisation des musées en île de France") +
  theme(plot.title = element_text(hjust = 0.5)) +
  coord_sf()
```



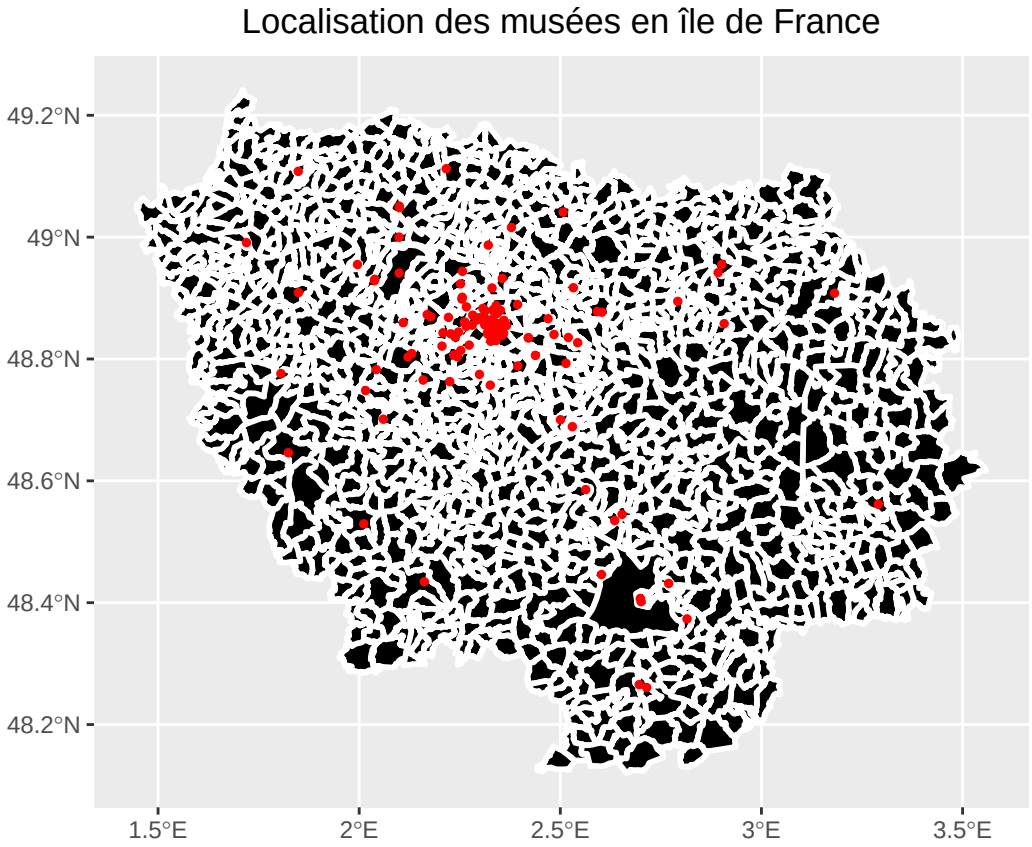

```
#Nous allons partitionner nos arrondissements de paris en deux parties pour un usage future
part1 <-AR[1:7,]
part2 <-AR[8:20,]
```

Opérations Géométriques

Affichage de plusieurs données sur une carte

Il est également possible d'afficher une carte avec plusieurs couches de données

```
ggplot() +
  geom_sf(data = IDF, size = 1, color = "white", fill = "black") +
  geom_sf(data = musees, size = 1, color = "red", fill = "black") +
  ggtitle("Localisation des musées en île de France") +
  theme(plot.title = element_text(hjust = 0.5)) +
  coord_sf()
```



Intersection de deux données

Elle permet de connaitre la zone qui fait objet d'intersection entre deux figures
Fonction: *st_intersection()*
Paramètres nécéessaires: *deux données (points, lignes, polygones)*

```
intersec = st_intersection(part1, AR)
```

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

```
## Warning: attribute variables are assumed to be spatially constant throughout all
## geometries
```

```
ggplot() +
  geom_sf(data = intersec, size = 1, color = "white", fill = "black") +
  ggtitle("Intersection") +
  theme(plot.title = element_text(hjust = 0.5))+
  coord_sf()
```

Nous remarquons juste que la carte affichée est celle de la partie 1 qui contient les 7 arrondissements choisis.

Union de deux cartes

Il permet de faire une union de cartes en une seule.
Nous faisons une union des deux parties des arrondissements de Paris que nous avions découper pour retrouver la carte de base qui comporte les arrondissements de Paris.

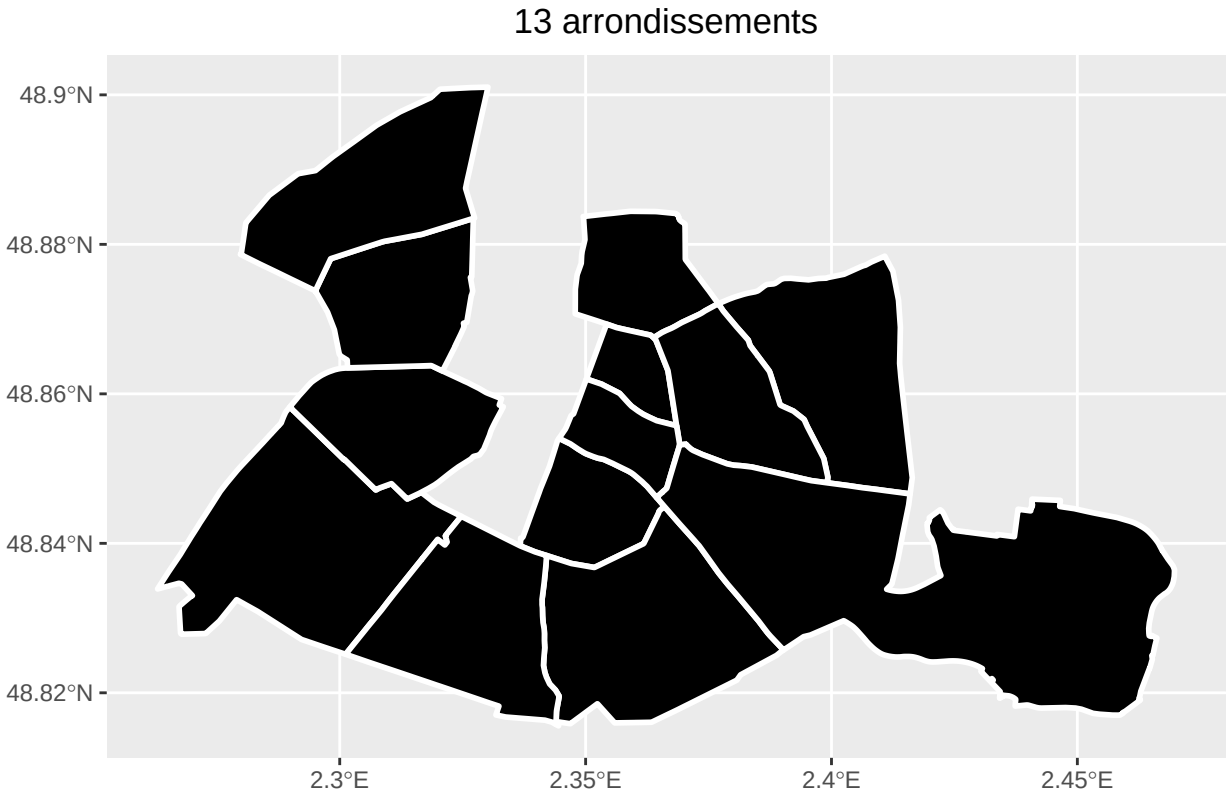
```
union <- st_union(part1,part2)
```

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

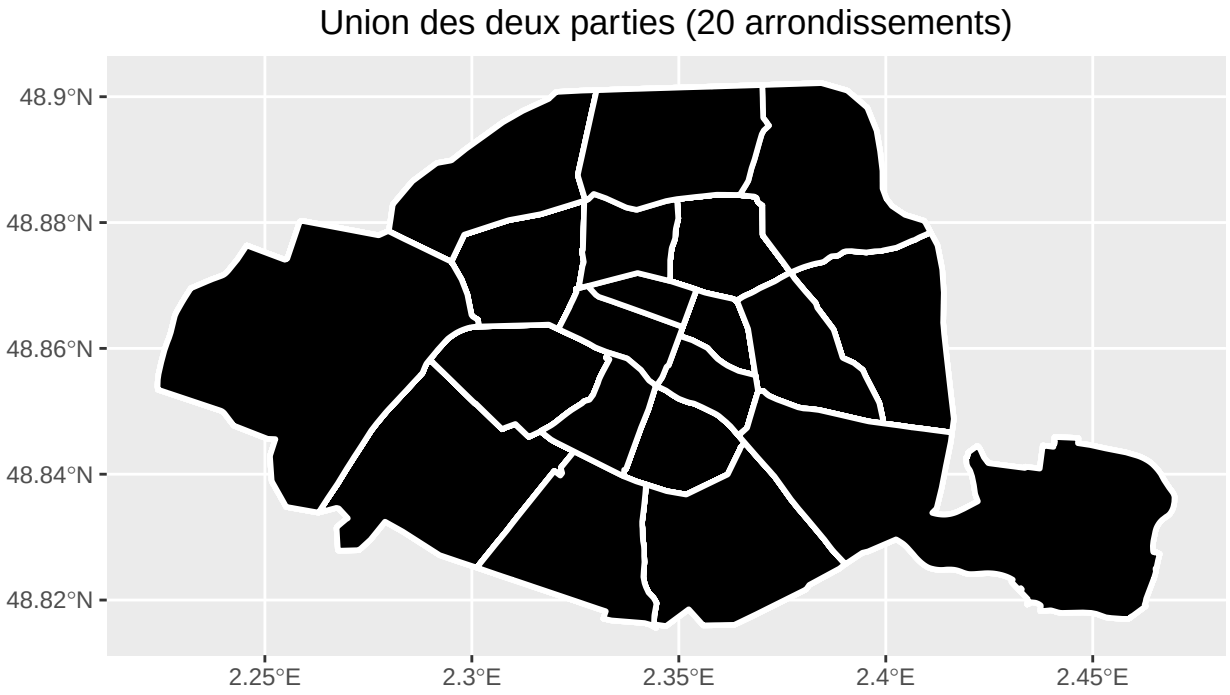
```
## Warning: attribute variables are assumed to be spatially constant throughout all
## geometries
```

```
#Partie 1: 7 arrondissements

ggplot() +
  geom_sf(data = part1, size = 1, color = "white", fill = "black") +
  ggtitle("7 arrondissements") +
  theme(plot.title = element_text(hjust = 0.5))+
  coord_sf()
```


```
#Partie 2: 13 arrondissements

ggplot() +
  geom_sf(data = part2, size = 1, color = "white", fill = "black") +
  ggtitle("13 arrondissements") +
  theme(plot.title = element_text(hjust = 0.5))+
  coord_sf()
```




```
#Carte de l'union des deux parties
ggplot() +
  geom_sf(data = union, size = 1, color = "white", fill = "black") +
  ggtitle("Union des deux parties (20 arrondissements)") +
  theme(plot.title = element_text(hjust = 0.5))+
  coord_sf()
```



Opératons MISC

Obtenir les coordonnées géographique d'un type de données

il permet d'obtenir les coordonnées géographique qui composent une donnée géométrique.

```
#Cas du premier arrondissement
coord <- st_coordinates(AR[1,])
coord[1:10,]
```

```
##              X        Y L1 L2
##  [1,] 2.328007 48.86992  1  1
##  [2,] 2.329966 48.86851  1  1
##  [3,] 2.330307 48.86836  1  1
##  [4,] 2.330657 48.86819  1  1
##  [5,] 2.331726 48.86795  1  1
##  [6,] 2.331726 48.86795  1  1
##  [7,] 2.333675 48.86752  1  1
##  [8,] 2.335869 48.86700  1  1
##  [9,] 2.335870 48.86700  1  1
## [10,] 2.337372 48.86665  1  1
```

Changer le type de donnée géométrique

Il est possible de transformer le type géométrique des données.

```r
test <- st_cast(AR, "LINESTRING")
```

```
## Warning in st_cast.sf(AR, "LINESTRING"): repeating attributes for all sub-
## geometries for which they may not be constant
```

```r
#Type de données d'origine (POLYGON)
AR$geometry
```

```
## Geometry set for 20 features
## geometry type:  POLYGON
## dimension:      XY
## bbox:           xmin: 2.224078 ymin: 48.81558 xmax: 2.469761 ymax: 48.90216
## geographic CRS: WGS 84
## First 5 geometries:

## POLYGON ((2.328007 48.86992, 2.329966 48.86851,...

## POLYGON ((2.351518 48.86443, 2.350949 48.86341,...

## POLYGON ((2.389429 48.90122, 2.390136 48.90108,...

## POLYGON ((2.344593 48.85405, 2.344284 48.85375,...

## POLYGON ((2.274268 48.87837, 2.277491 48.87796,...
```

```r
#Type de données après application de la fonction st_cast (LINESTRING)
test$geometry
```

```
## Geometry set for 20 features
## geometry type:  LINESTRING
## dimension:      XY
## bbox:           xmin: 2.224078 ymin: 48.81558 xmax: 2.469761 ymax: 48.90216
## geographic CRS: WGS 84
## First 5 geometries:

## LINESTRING (2.328007 48.86992, 2.329966 48.8685...

## LINESTRING (2.351518 48.86443, 2.350949 48.8634...

## LINESTRING (2.389429 48.90122, 2.390136 48.9010...

## LINESTRING (2.344593 48.85405, 2.344284 48.8537...

## LINESTRING (2.274268 48.87837, 2.277491 48.8779...
```

Fonction de mesures géométrique

Il s'agit des fonctions qui permettent de connaitre les mesures pour un objet géométrique ou entre deux ou plusieurs de ces objets. Ces mesures peuvent être par exemple la distance, la surface, la longueur...

Surface d'un polygone

Permet de calculer la surface d'une géométrie de polygone en fonction du système de référence de coordonnées actuel. La mesure est en m^2

```
surface <- (st_area(AR))
paste("La surface de l'arrondissement " ,AR$l_aroff, " est ", surface, " m^2")
```

```
##  [1] "La surface de l'arrondissement  Louvre  est  1824989.07793839  m^2"
##  [2] "La surface de l'arrondissement  Bourse  est  991349.84192924  m^2"
##  [3] "La surface de l'arrondissement  Buttes-Chaumont  est  6793807.74783465  m^2"
##  [4] "La surface de l'arrondissement  Luxembourg  est  2153581.51605498  m^2"
##  [5] "La surface de l'arrondissement  Passy  est  16375969.114879  m^2"
##  [6] "La surface de l'arrondissement  Opéra  est  2178705.89611031  m^2"
##  [7] "La surface de l'arrondissement  Buttes-Montmartre  est  5997023.79307874  m^2"
##  [8] "La surface de l'arrondissement  Entrepôt  est  2892278.33836847  m^2"
##  [9] "La surface de l'arrondissement  Vaugirard  est  8497022.28420066  m^2"
## [10] "La surface de l'arrondissement  Temple  est  1171123.70278399  m^2"
## [11] "La surface de l'arrondissement  Palais-Bourbon  est  4090938.5276088  m^2"
## [12] "La surface de l'arrondissement  Panthéon  est  2539964.93489387  m^2"
## [13] "La surface de l'arrondissement  Élysée  est  3880778.8344164  m^2"
## [14] "La surface de l'arrondissement  Gobelins  est  7151138.39502654  m^2"
## [15] "La surface de l'arrondissement  Batignolles-Monceau  est  5669797.66587133  m^2"
## [16] "La surface de l'arrondissement  Ménilmontant  est  5984671.81437322  m^2"
## [17] "La surface de l'arrondissement  Popincourt  est  3666215.95926502  m^2"
## [18] "La surface de l'arrondissement  Hôtel-de-Ville  est  1600934.80432109  m^2"
## [19] "La surface de l'arrondissement  Observatoire  est  5616305.5305404  m^2"
## [20] "La surface de l'arrondissement  Reuilly  est  16318797.9638278  m^2"
```

Nous remarquons que les résultats sont appromatifs comparés aux résultats fournis dans le dataset.

Connaitre la distance entre deux points

Pour cette partie nous allons utiliser les données relatives aux musées pour trouver les

```
#Effectuons le même test sur les données de musées.

dis <- st_distance(musees[1,], musees[2,])
paste("La distance entre ",musees[1,]$nom_du_muse , " ET ", musees[2,]$nom_du_muse, " EST ", dis, " mèt
```

```
## [1] "La distance entre  Musée du Maréchal Leclerc de Hautecloque et de la Libération de Paris - Musé
```

Il est à noter que la distance se mesure par défaut en *dégré* sous le package SF

Fonction de Vérification

Il s'agit des fonctions qui permettent de vérifier la véracité d'une hypothèse. Ils peuevent être aussi être considérés comme des fonctions de logique.

Vérifier si une figure contient une autre

Souvent utilisée pour vérifier si un point est inclus dans un polygone.

```
#Nous vérifions si les 7 premiers arrondissements de paris sont dans Paris (logique quand même)...
cont <- st_contains(part1, AR)
```

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

```
cont
```

```
## Sparse geometry binary predicate list of length 7, where the predicate was 'contains'
##  1: 1
##  2: 2
##  3: 3
##  4: 4
##  5: 5
##  6: 6
##  7: 7
```

Les résultats nous montre que tous les éléments de la partie1 (7 premiers arrondissements) sont inclus avec leurs correspondances d'où la confirmation de notre hypothèse. Notons que pour les partie qui n'auraient pas été inclues, nous aurions eu des résultats en *empty*
Exemple:

```
#Nous vérifions si les 7 premiers arrondissements de Paris sont inclus dans les 13 derniers
cont2 <- st_contains(part1, part2)
```

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

```
cont2
```

```
## Sparse geometry binary predicate list of length 7, where the predicate was 'contains'
##  1: (empty)
##  2: (empty)
##  3: (empty)
##  4: (empty)
##  5: (empty)
##  6: (empty)
##  7: (empty)
```

Vérifier si deux figures se partagent la même géométrie

```
confirm <- st_equals(part1, AR)
confirm
```

```
## Sparse geometry binary predicate list of length 7, where the predicate was 'equals'
##  1: 1
##  2: 2
```

```
##  3: 3
##  4: 4
##  5: 5
##  6: 6
##  7: 7
```

Nous obtenons un résultat positif.

Vérifier une intersection entre deux figures

Nous notons que cette fonction est différente de la fonction *intersection* qui retourne une figure géométrique que nous avons vu un peu plus haut dans le document.

```
inter <- st_intersects(AR, IDF)
```

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

```
inter
```

```
## Sparse geometry binary predicate list of length 20, where the predicate was 'intersects'
## first 10 elements:
##  1: 1172, 1173, 1179, 1180
##  2: 1173, 1180, 1181
##  3: 1050, 1099, 1105, 1109, 1189, 1190
##  4: 1172, 1177, 1178
##  5: 1007, 1008, 1009, 1011, 1032, 1179, 1187, 1188, 1192
##  6: 1179, 1180, 1181, 1188, 1189
##  7: 1043, 1050, 1094, 1188, 1189, 1190
##  8: 1180, 1181, 1189, 1190
##  9: 1015, 1177, 1178, 1185, 1186, 1187
##  10: 1172, 1173, 1174, 1181, 1182
```

Nous remarquons que la liste nous montre les différents arrondissements qui s'intercèdent avec les communes de l'île de France.

Vérifier si un Polygone est entièrement inclus dans un autre

Nous considérons deux cas:

```
couvre1<- st_covered_by(part1, AR)
```

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

```
couvre2<- st_covered_by(part1, part2)
```

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

```r
#Cas où il y a une couverture totale
couvre1
```

```
## Sparse geometry binary predicate list of length 7, where the predicate was 'covered_by'
##  1: 1
##  2: 2
##  3: 3
##  4: 4
##  5: 5
##  6: 6
##  7: 7
```

```r
#Cas où nous n'avons pas de couverture
couvre2
```

```
## Sparse geometry binary predicate list of length 7, where the predicate was 'covered_by'
##  1: (empty)
##  2: (empty)
##  3: (empty)
##  4: (empty)
##  5: (empty)
##  6: (empty)
##  7: (empty)
```

CheatSheet de SF:

Ceci est une image de la synthèse des fonctions principales du package *SF*

Spatial manipulation with sf: : CHEAT SHEET

The sf package provides a set of tools for working with geospatial vectors, i.e. points, lines, polygons, etc.

Geometric confirmation

st_contains(x, y, ...) Identifies if x is within y (i.e. point within polygon)

st_covered_by(x, y, ...) Identifies if x is completely within y (i.e. polygon completely within polygon)

st_covers(x, y, ...) Identifies if any point from x is outside of y (i.e. polygon outside polygon)

st_crosses(x, y, ...) Identifies if any geometry of x have commonalities with y

st_disjoint(x, y, ...) Identifies when geometries from x do not share space with y

st_equals(x, y, ...) Identifies if x and y share the same geometry

st_intersects(x, y, ...) Identifies if x and y geometry share any space

st_overlaps(x, y, ...) Identifies if geometries of x and y share space, are of the same dimension, but are not completely contained by each other

st_touches(x, y, ...) Identifies if geometries of x and y share a common point but their interiors do not intersect

st_within(x, y, ...) Identifies if x is in a specified distance to y

Geometric operations

st_boundary(x) Creates a polygon that encompasses the full extent of the geometry

st_buffer(x, dist, nQuadSegs) Creates a polygon covering all points of the geometry within a given distance

st_centroid(x, ..., of_largest_polygon) Creates a point at the geometric centre of the geometry

st_convex_hull(x) Creates geometry that represents the minimum convex geometry of x

st_line_merge(x) Creates linestring geometry from sewing multi linestring geometry together

st_node(x) Creates nodes on overlapping geometry where nodes do not exist

st_point_on_surface(x) Creates a point that is guarenteed to fall on the surface of the geometry

st_polygonize(x) Creates polygon geometry from linestring geometry

st_segmentize(x, dfMaxLength, ...) Creates linesting geometry from x based on a specified length

st_simplify(x, preserveTopology, dTolerance) Creates a simplified version of the geometry based on a specified tolerance

Geometry creation

st_triangulate(x, dTolerance, bOnlyEdges) Creates polygon geometry as triangles from point geometry

st_voronoi(x, envelope, dTolerance, bOnlyEdges) Creates polygon geometry covering the envolope of x, with x at the centre of the geometry

st_point(x, c(numeric vector), dim = "XYZ") Creating point geometry from numeric values

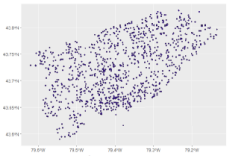
st_multipoint(x = matrix(numeric values in rows), dim = "XYZ") Creating multi point geometry from numeric values

st_linestring(x = matrix(numeric values in rows), dim = "XYZ") Creating linestring geometry from numeric values

st_multilinestring(x = list(numeric matricesin rows), dim = "XYZ") Creating multi linestring geometry from numeric values

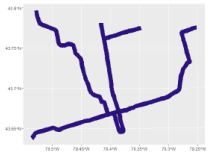
st_polygon(x = list(numeric matrices in rows), dim = "XYZ") Creating polygon geometry from numeric values

st_multipolygon(x = list(numeric matrices in rows), dim = "XYZ") Creating multi polygon geometry from numeric values



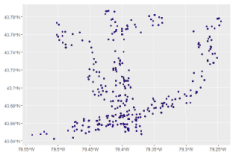
ggplot() +
geom_sf(data = schools)

+



ggplot() +
geom_sf(data = subway)

=>



ggplot() +
geom_sf(data = st_intersection(schools, st_buffer(subway, 1000)))

This cheatsheet presents the sf package [Edzer Pebesma 2018] in version 0.6.3. See https://github.com/r-spatial/sf for more details.

Références

- https://www.data.gouv.fr/fr/datasets/liste-des-musees-franciliens-idf/#_
- https://stackoverflow.com/questions/40675778/center-plot-title-in-ggplot2
- https://r-spatial.github.io/sf/reference/st_as_sf.html
- https://github.com/rstudio/cheatsheets/blob/master/sf.pdf
- https://www.data.gouv.fr/fr/datasets/arrondissements-1/#_
- https://www.data.gouv.fr/fr/datasets/apur-communes-ile-de-france/