R pour le Géospatial

Transform

Etienne Racine

Construire des géométries

```
giza <- tribble(
  ~what, ~longitude, ~latitude,
  "Giza", 31.1342, 29.9792,
  "Khafre", 31.130833, 29.976111,
  "Menkaure", 31.128333, 29.9725,
  "Khentkaus I", 31.135608, 29.973406,
  "Sphynx", 31.137778, 29.975278
)
```

what	longitude	latitude
Giza	31.13420	29.97920
Khafre	31.13083	29.97611
Menkaure	31.12833	29.97250
Khentkaus I	31.13561	29.97341
Sphynx	31.13778	29.97528

st_as_sf

```
giza %>%
  mapview()
#> Error: oops! Arguments xcol and/or ycol are missing! You probably expected .
#> to be a spatial object. However it is of class tbl_df. Either convert . to a
#> spatial object or provide xcol and ycol.
```

(Dans ce cas-ci, c'est possible d'utiliser `mapview` enspécifiant `xcol = longitude, ycol = latitude`)

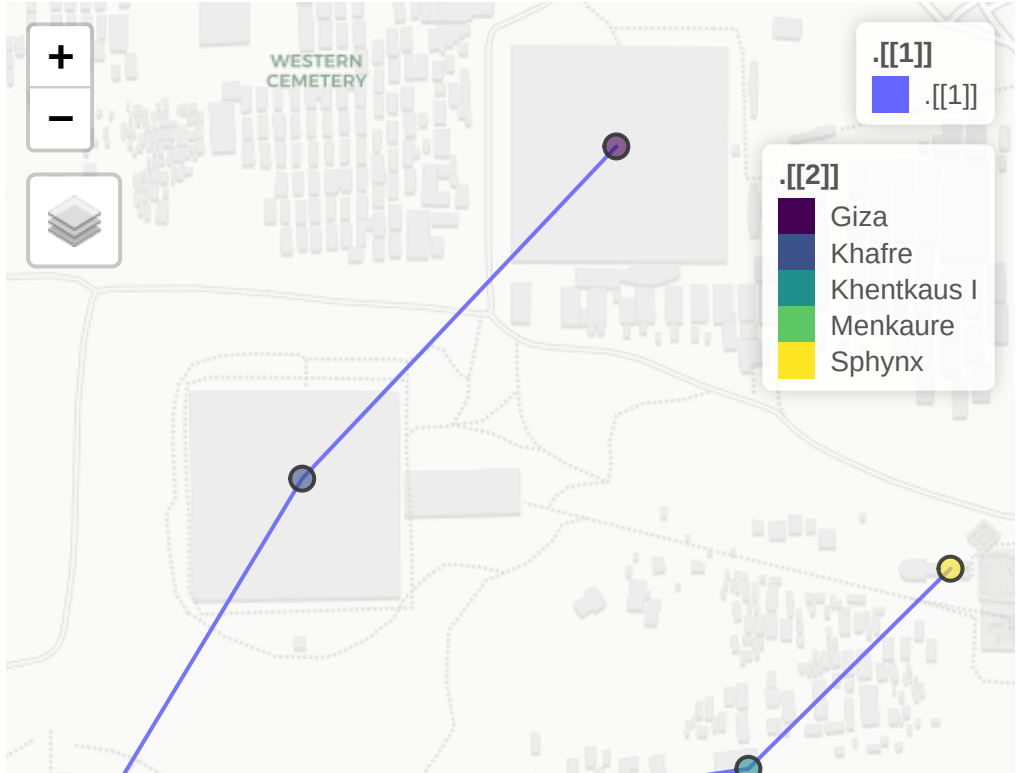
```r
library(sf)
giza <- giza %>%
  st_as_sf(coords = c("longitude", "latitude"), crs = 4326)
```

```
## Simple feature collection with 5 features and 1 field
## geometry type:  POINT
## dimension:      XY
## bbox:           xmin: 31.12833 ymin: 29.9725 xmax: 31.13778 ymax: 29.9792
## epsg (SRID):    4326
## proj4string:    +proj=longlat +datum=WGS84 +no_defs
## # A tibble: 5 x 2
##   what                  geometry
##   <chr>              <POINT [°]>
## 1 Giza          (31.1342 29.9792)
## 2 Khafre      (31.13083 29.97611)
## 3 Menkaure     (31.12833 29.9725)
## 4 Khentkaus I (31.13561 29.97341)
## 5 Sphynx      (31.13778 29.97528)
```

Créer des lignes

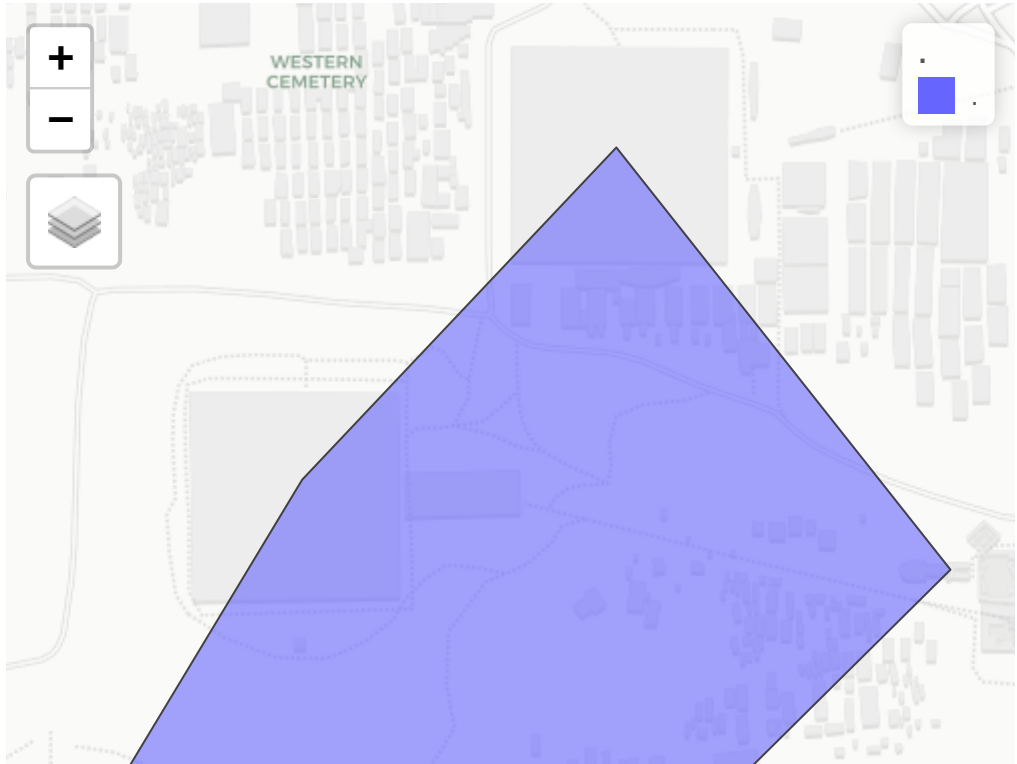
```
links <- giza %>%
  summarise(st_cast(st_combine(geometry), "LINESTRING"))

list(links, giza) %>%
  mapview()
```



Créer des polygones

```
polygon <- giza %>%
  slice(c(1:5, 1)) %>%
  summarise(geometry = st_cast(st_combine(geometry), "POLYGON"))
polygon %>%
  mapview()
```



Well-known-text (WKT)

```
st_as_text(polygon$geometry)
```

```
## [1] "POLYGON ((31.1342 29.9792, 31.13083 29.97611, 31.12833 29.9725, 31.13561 29.97341, 31.13
```

Well-known-text (WKT)

```
# Full circle WKT
polygon %>%
  mutate(text = st_as_text(polygon$geometry)) %>%
  st_as_sf(wkt = text)
```

```
## Simple feature collection with 1 feature and 1 field
## geometry type:  POLYGON
## dimension:      XY
## bbox:           xmin: 31.12833 ymin: 29.9725 xmax: 31.13778 ymax: 29.9792
## epsg (SRID):    4326
## proj4string:    +proj=longlat +datum=WGS84 +no_defs
##                         geometry
## 1 POLYGON ((31.1342 29.9792, ...
##
## 1 POLYGON ((31.1342 29.9792, 31.13083 29.97611, 31.12833 29.9725, 31.13561 29.97341, 31.13778
```

GEOMETRY

On a vu des points, des lignes et des polygones et voici un autre type important:

```
GEOMETRYCOLLECTION(POINT(2 0),POLYGON((0 0, 1 0, 1 1, 0 1, 0 0)))
```

À vous

- Pensez à trois emplacements importants sur la terre
- Utilisez openstreetmap.org et/ou google.com/maps pour obtenir les coordonnées
- Transformez les coordonnées en géométries (hint: `tribble`) et `plot`
- Essayez avec `mapedit()`. Laquelle préférez-vous?

Propriétés des géométries

```
polygon %>%
  mutate(
    area = st_area(geometry),
    perimeter = st_length(geometry),
    #vertex = geotidy::st_npoints(geometry)
  )
```

```
## Simple feature collection with 1 feature and 2 fields
## geometry type:  POLYGON
## dimension:      XY
## bbox:           xmin: 31.12833 ymin: 29.9725 xmax: 31.13778 ymax: 29.9792
## epsg (SRID):    4326
## proj4string:    +proj=longlat +datum=WGS84 +no_defs
##                         geometry           area    perimeter
## 1 POLYGON ((31.1342 29.9792, ... 337380.2 [m^2] 2498.748 [m]
```

C'est aussi possible d'extraire les coordonnées d'une géométrie.

```
st_coordinates(polygon$geometry)
```

```
##              X        Y L1 L2
## [1,] 31.13420 29.97920  1  1
## [2,] 31.13083 29.97611  1  1
## [3,] 31.12833 29.97250  1  1
## [4,] 31.13561 29.97341  1  1
## [5,] 31.13778 29.97528  1  1
## [6,] 31.13420 29.97920  1  1
```

Et de géométrie vers points.

```
polygon %>%
  mutate(centroid = st_centroid(geometry))
```

```
## Warning in st_centroid.sfc(geometry): st_centroid does not give correct
## centroids for longitude/latitude data
```

```
## Simple feature collection with 1 feature and 0 fields
## Active geometry column: geometry
## geometry type:  POLYGON
## dimension:      XY
## bbox:           xmin: 31.12833 ymin: 29.9725 xmax: 31.13778 ymax: 29.9792
## epsg (SRID):    4326
## proj4string:    +proj=longlat +datum=WGS84 +no_defs
##                         geometry                  centroid
## 1 POLYGON ((31.1342 29.9792, ... POINT (31.13336 29.97532)
```

À vous

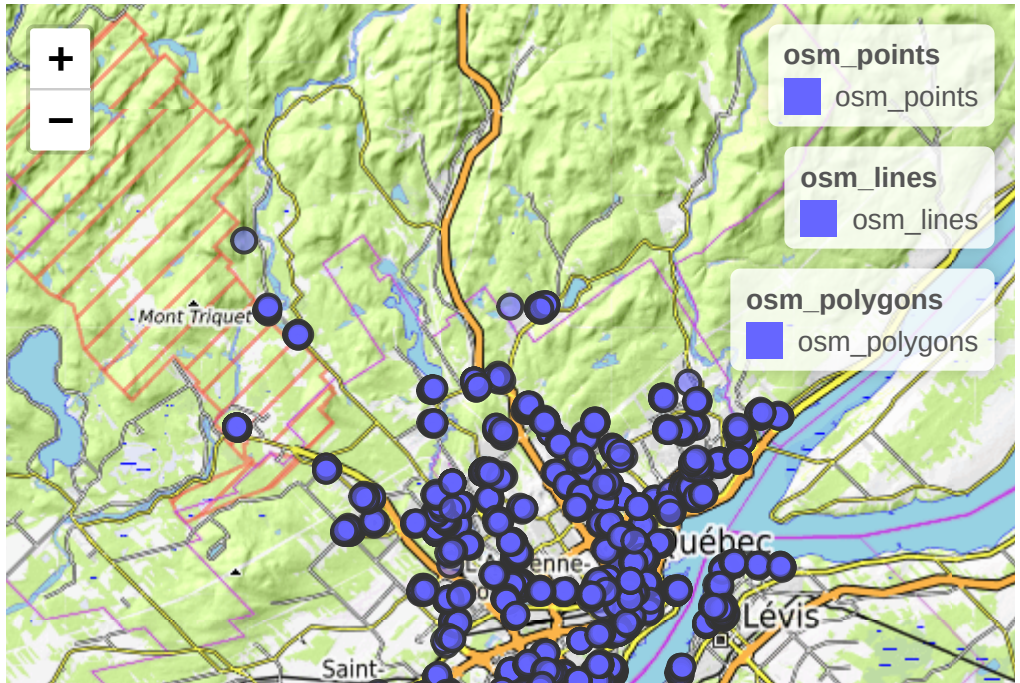
- Approximez la longueur de la frontière entre les États-Unis et le Mexique en utilisant `mapedit::editMap()`.
- Nous avons utilisé `st_combine` pour construire un polygone à partir de points. Quelle est la différence entre `st_combine` et `st_union`?
- Quelle est la superficie de votre aire d'étude
- Quelle est la longueur totale des rues dans votre aire d'étude? Indice: regardez la colonne `highway`.
- Quelle est la proportion de parcs dans votre aire d'étude? Indice: `leisure=park`.

Relations

Obtenir les données

```
school_data <- opq("Quebec City") %>%
  add_osm_feature(key = "amenity", value = "school") %>%
  osmdata_sf()

school_data %>%
  .[c(4, 5, 6, 8)] %>% # osm_* elements
  mapview()
```



Écoles avec un hopital à moins de 3km.

```
hospital <- opq("Quebec City") %>%
  add_osm_feature(key = "amenity", value = "hospital") %>%
  osmdata_sf() %>%
  .$osm_polygons
```

```
list(hospital = hospital, school = school) %>%
  mapview()
```

```r
school <- school %>%
  select(name) %>%
  mutate(n_hospital = st_is_within_distance(geometry, hospital, 3000) %>% map_int(length))

as_tibble(school)
```

```
## # A tibble: 21 x 3
##    name                                          geometry n_hospital
##    <chr>                                    <POINT [°]>      <int>
##  1 Centre du Nouvel-Horizon                (-71.20917 46.85076)          4
##  2 École Secondaire Joseph-François-Perra… (-71.22899 46.80608)          9
##  3 École Secondaire Mont-Saint-Sacrement   (-71.50208 46.97832)          0
##  4 École de la Passerelle                   (-71.37272 46.9085)          0
##  5 École de la Boisé                       (-71.34218 46.90985)          1
##  6 Patinoire Bon Pasteur                   (-71.30893 46.89907)          1
##  7 Centre de formation professionnelle Ma… (-71.29594 46.76747)          3
##  8 Maison des Adultes                      (-71.25488 46.85683)          4
##  9 École Joseph-Paquin                     (-71.25437 46.85634)          4
## 10 CFP Samuel-De Champlain                    (-71.20854 46.86)          4
## # … with 11 more rows
```

`st_is_within_distance` retourne un format peu commun.

Tampon

Les tampons sont pratiques pour représenter des distances sur une carte.

```
school <-
  school %>%
  mutate(buffer_3km = geometry %>%
           st_transform(3978) %>%
           st_buffer(3000) %>% #<<
           st_transform(4326)
         )

list(
  hospital = hospital,
  school = school,
  "3 km" = school$buffer_3km
  ) %>%
  mapview()
```

Dans la prochaine section on discutera de l'opération `st_transform`. On peut les ignorer en attendant.

À vous

- Sélectionnez les écoles dans votre aire d'étude.
- Pouvez-vous ajouter les écoles manquantes qui sont des polygones? (indice: `st_centroid`)
- Pouvez-vous colorier les tampons en fonction de la présence ou l'absence d'un hôpital?

Coordinate Reference System

Synonymes

- Coordinate Reference System (CRS)
- Spatial Reference IDentifier (SRID)
- European Petroleum Survey Group (EPSG)

Angulaire vs cartésienne

```
school %>% as_tibble()
```

```
## # A tibble: 21 x 4
##    name                     geometry n_hospital                 buffer_3km
##    <chr>                 <POINT [°]>      <int>               <POLYGON [°]>
##  1 Centre …     (-71.20917 46.85076)          4 ((-71.17288 46.84098, -71.…
##  2 École S…     (-71.22899 46.80608)          9 ((-71.19274 46.79631, -71.…
##  3 École S…     (-71.50208 46.97832)          0 ((-71.46562 46.96865, -71.…
##  4 École d…      (-71.37272 46.9085)          0 ((-71.33635 46.89878, -71.…
##  5 École d…     (-71.34218 46.90985)          1 ((-71.30582 46.90012, -71.…
##  6 Patinoi…     (-71.30893 46.89907)          1 ((-71.27259 46.88932, -71.…
##  7 Centre …     (-71.29594 46.76747)          3 ((-71.2597 46.75773, -71.2…
##  8 Maison …     (-71.25488 46.85683)          4 ((-71.21858 46.84707, -71.…
##  9 École J…     (-71.25437 46.85634)          4 ((-71.21807 46.84658, -71.…
## 10 CFP Sam…        (-71.20854 46.86)          4 ((-71.17225 46.85022, -71.…
## # … with 11 more rows
```

Reprojection

```
school %>%
  mutate(plannar = st_transform(geometry, 3978)) %>%
  as_tibble()
```

```
## # A tibble: 21 x 5
##    name                  geometry n_hospital               buffer_3km
##    <chr>              <POINT [°]>      <int>             <POLYGON [°]>
##  1 Cent…      (-71.20917 46.85076)          4 ((-71.17288 46.84098, -7…
##  2 Écol…      (-71.22899 46.80608)          9 ((-71.19274 46.79631, -7…
##  3 Écol…      (-71.50208 46.97832)          0 ((-71.46562 46.96865, -7…
##  4 Écol…       (-71.37272 46.9085)          0 ((-71.33635 46.89878, -7…
##  5 Écol…      (-71.34218 46.90985)          1 ((-71.30582 46.90012, -7…
##  6 Pati…      (-71.30893 46.89907)          1 ((-71.27259 46.88932, -7…
##  7 Cent…      (-71.29594 46.76747)          3 ((-71.2597 46.75773, -71…
##  8 Mais…      (-71.25488 46.85683)          4 ((-71.21858 46.84707, -7…
##  9 Écol…      (-71.25437 46.85634)          4 ((-71.21807 46.84658, -7…
## 10 CFP …         (-71.20854 46.86)          4 ((-71.17225 46.85022, -7…
## # … with 11 more rows, and 1 more variable: plannar <POINT [m]>
```

WGS84

```
giza <- giza %>%
  st_as_sf(coords = c("longitude", "latitude"), crs = 4326)
```

UTM

6° zones

```
library(leaflet)
leaflet() %>%
  addTiles() %>%
  setView(0, 0, 2) %>%
  addGraticule(interval = 6, style = list(color = "#FF0000", weight = 1))
```

Set CRS

```
giza$geometry %>%
  st_crs()
```

```
## Coordinate Reference System:
##   EPSG: 4326
##   proj4string: "+proj=longlat +datum=WGS84 +no_defs"
```

```
# remove crs
giza$geometry %>%
  st_set_crs(NA)
```

```
## Geometry set for 5 features
## geometry type:  POINT
## dimension:      XY
## bbox:           xmin: 31.12833 ymin: 29.9725 xmax: 31.13778 ymax: 29.9792
## epsg (SRID):    NA
## proj4string:    NA
```

```
## POINT (31.1342 29.9792)
```

```r
# remove crs and set it back to 4326
giza$geometry %>%
  st_set_crs(NA) %>%
  st_set_crs(4326)
```

```
## Geometry set for 5 features
## geometry type:  POINT
## dimension:      XY
## bbox:           xmin: 31.12833 ymin: 29.9725 xmax: 31.13778 ymax: 29.9792
## epsg (SRID):    4326
## proj4string:    +proj=longlat +datum=WGS84 +no_defs
```

```
## POINT (31.1342 29.9792)
```

```
## POINT (31.13083 29.97611)
```

```
## POINT (31.12833 29.9725)
```

```
## POINT (31.13561 29.97341)
```

```
## POINT (31.13778 29.97528)
```

À vous

- Quelle système de coordonnées utilise `osmdata`?
- Quelles est la zone UTM de votre aire d'étude? Vous pouvez utiliser un service comme https://www.latlong.net/lat-long-utm.html

Risques

- Utiliser une projection non désirée (`sf` tente de nous aider).
- Utiliser des systèmes différents en comparant des données (idéalement ça bloque).