First steps in spatial data handling and visualization

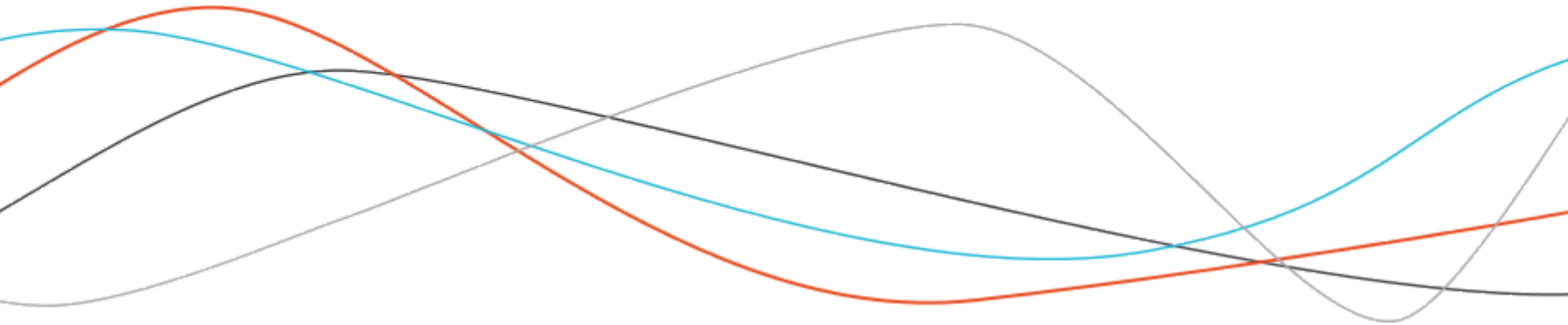
useR! 2020 - 21 July - remote

Sébastien Rochette, Dorris Scott, Jakub Nowosad

Material of this course is on Github: statnmap/user2020_rspatial_tutorial

Introduction

Presentation of the tutorial



Overview

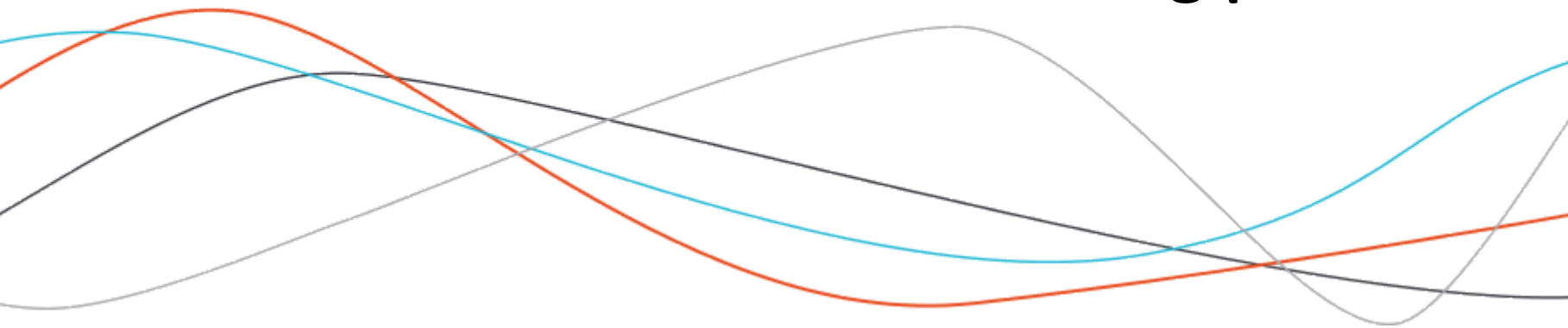

useR!
First steps in spatial data handling and visualization
Sébastien Rochette
Dorris Scott
Jakub Nowosad
July 27th
14:00 UTC
AfricaR
THINKR
Kampala
NairobiR

Planning

- **3 instructors**: Sébastien Rochette, Dorris Scott, Jakub Nowosad
- **3 facilitators**: Shelmith Kariuki, Christopher Maronga, Mark Okello
- **3 hours tutorial**
- **3 parts**:
  - Tutorial presentation, Draw maps with {tmap}, What are spatial data?
  - Read/write spatial vector data, Manipulate vector data with the {tidyverse}
  - Manipulate and intersect spatial data
- **3 times 1 hour**: ~33' presentation + 2' quizz + 20' exercises + 5' break
- **1 repository**: https://github.com/statnmap/user2020_rspatial_tutorial

What is CRUZ?

The ThinkR e-learning platform



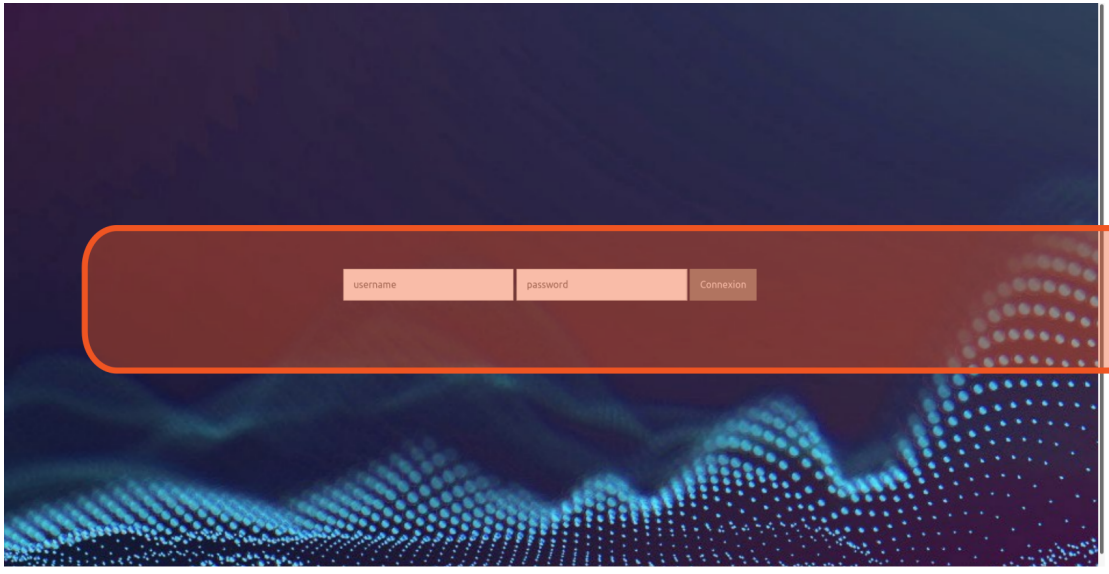
Take a tour!

CRUZ is the home-made ThinkR e-learning platform

- Everything you need during the course is available in one place
- The server has all dependencies required for your tutorial
- You can communicate directly with your instructors

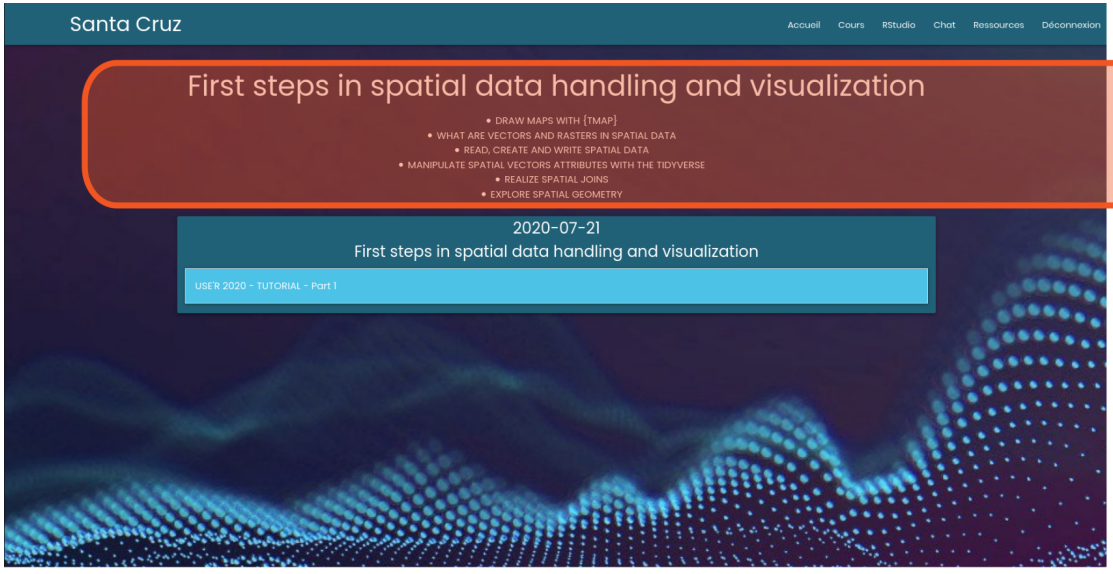
Please take this tour to get used to the platform

Connection



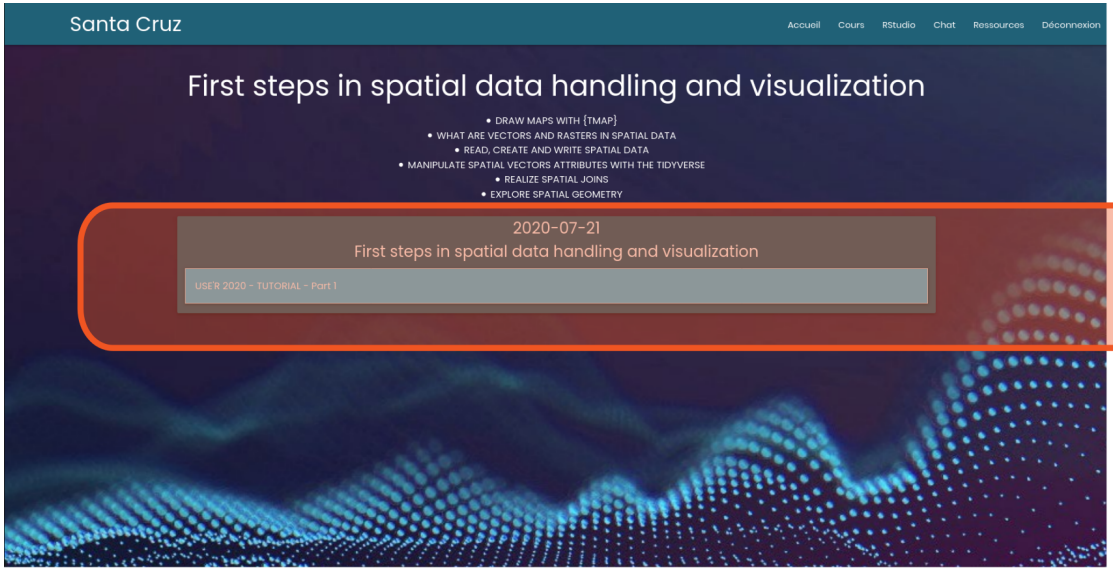
Connection
» Use your credentials to connect

Home page



Presentation of the learning objectives

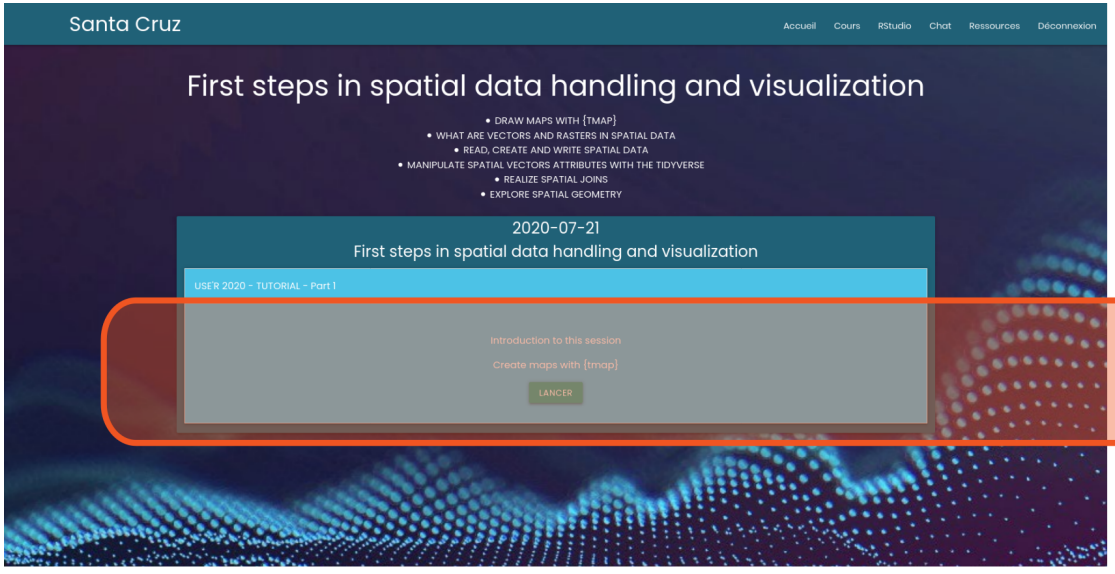
Home page



Chapters

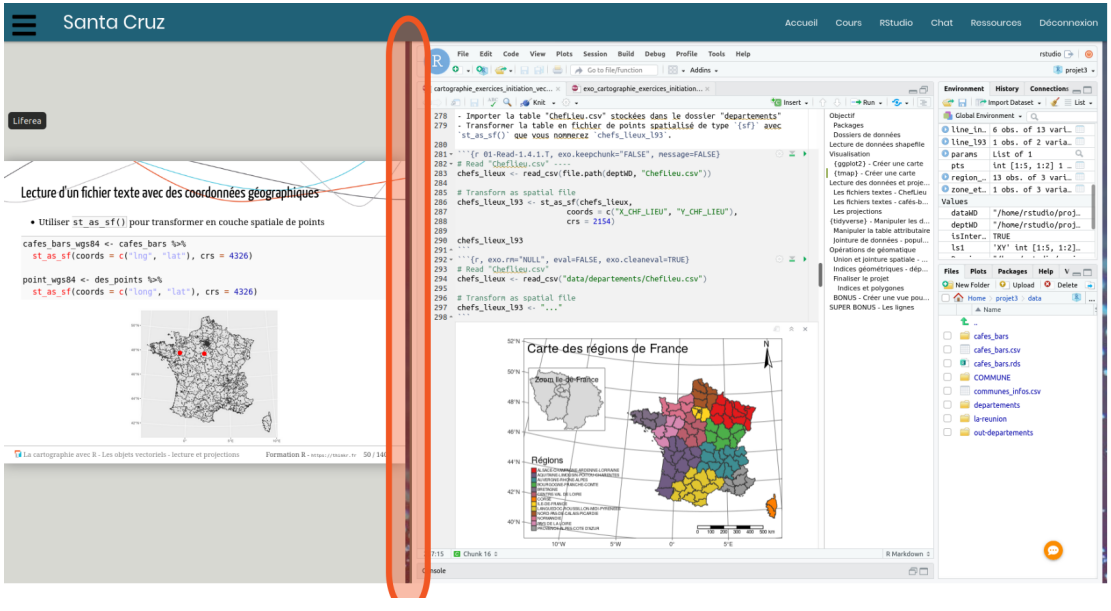
» Click on the inside box to reveal the "launch" button

Home page



Launch the tutorial
» *Click on the "launch" ("lancer") button*

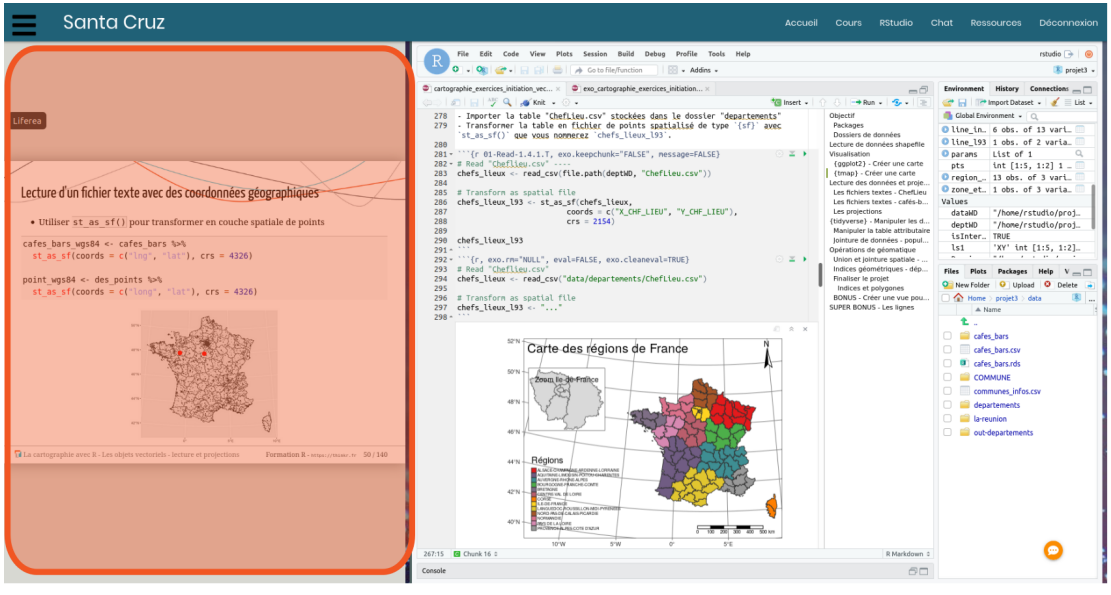
Pratice



Vertical separator

» *Maintain left click and use the mouse the resize the window*

Pratice

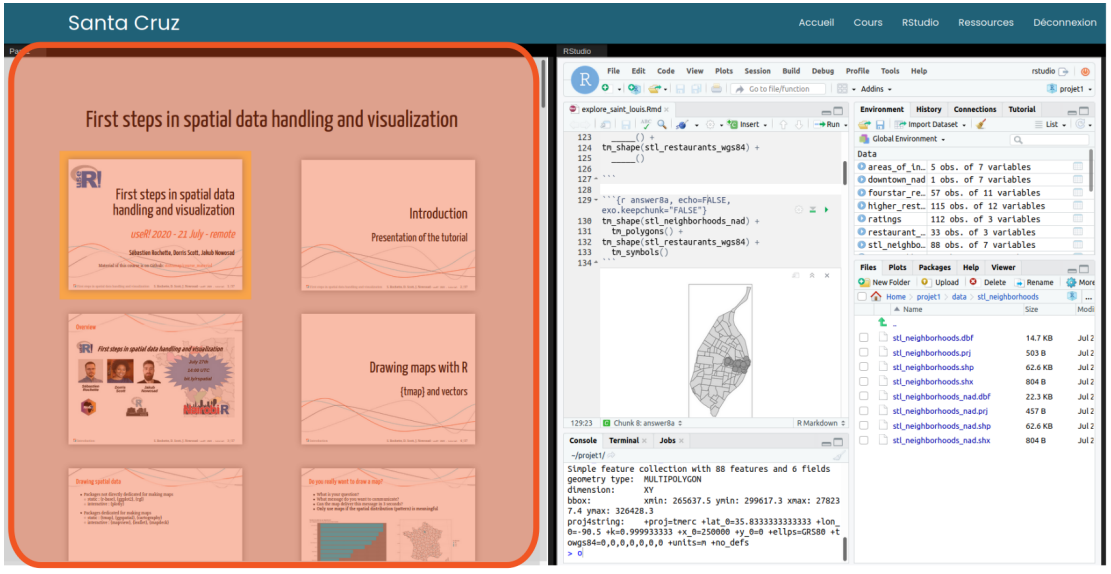


Slides

» Use arrows of your keyboard to go up and down in the slides

» You can copy-paste the code parts

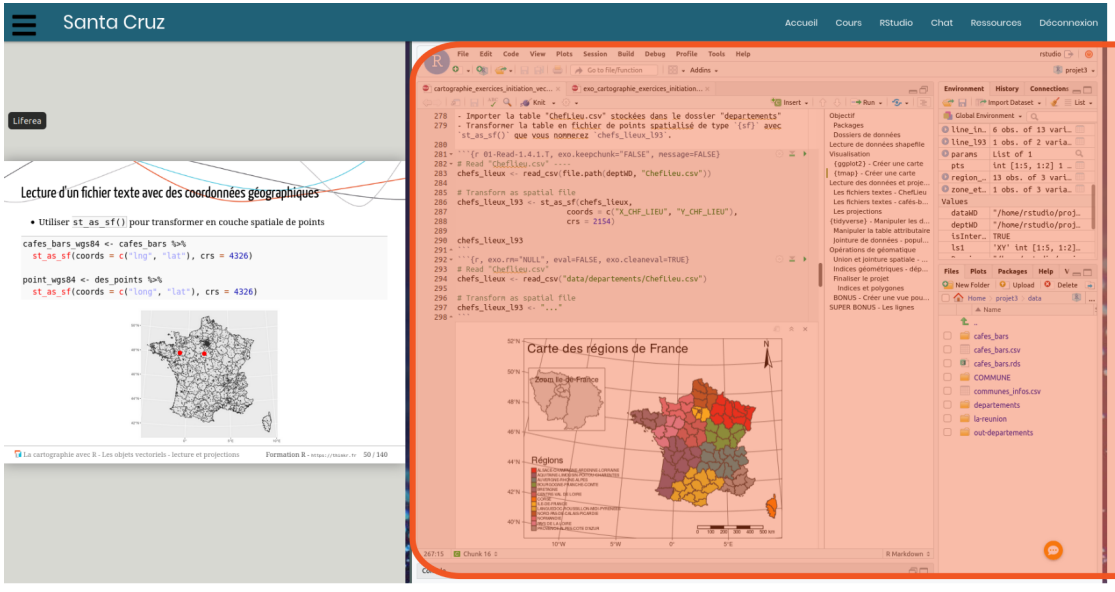
Pratice



Slides

» *Press "o" (like Overview) to navigate through slides.*
» *Click on a slide to come back to normal mode*

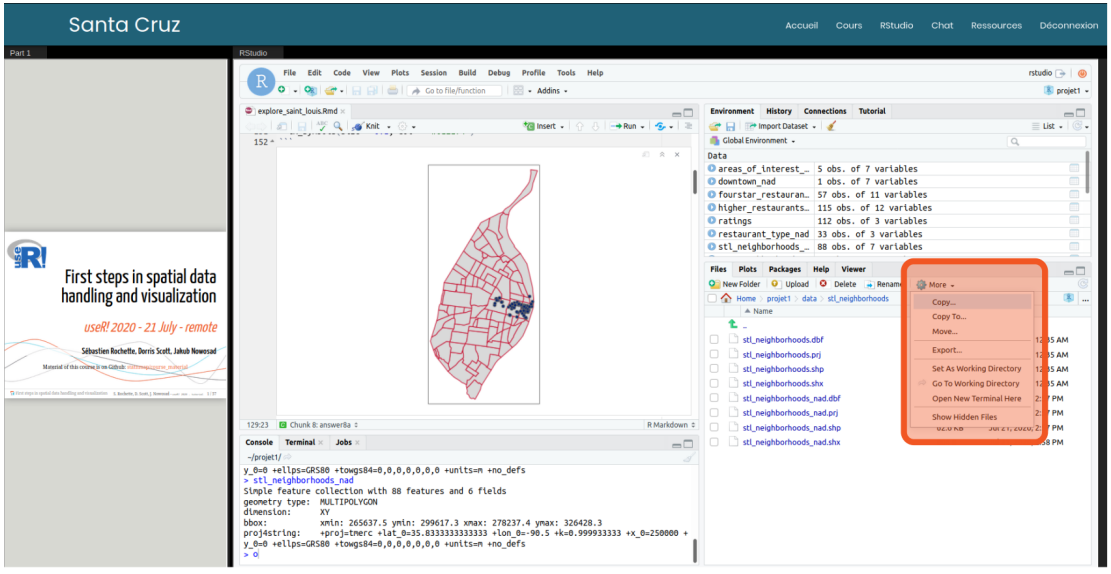
Pratice



RStudio Server

RStudio project opened with your data and Rmd files for exercises
» Create a new Rmd file to take your notes (if you know how to do it)

Export



Export button

This allows to download the content of the server on your own computer

» Select one or more file(s) in the file pane using the checkboxes

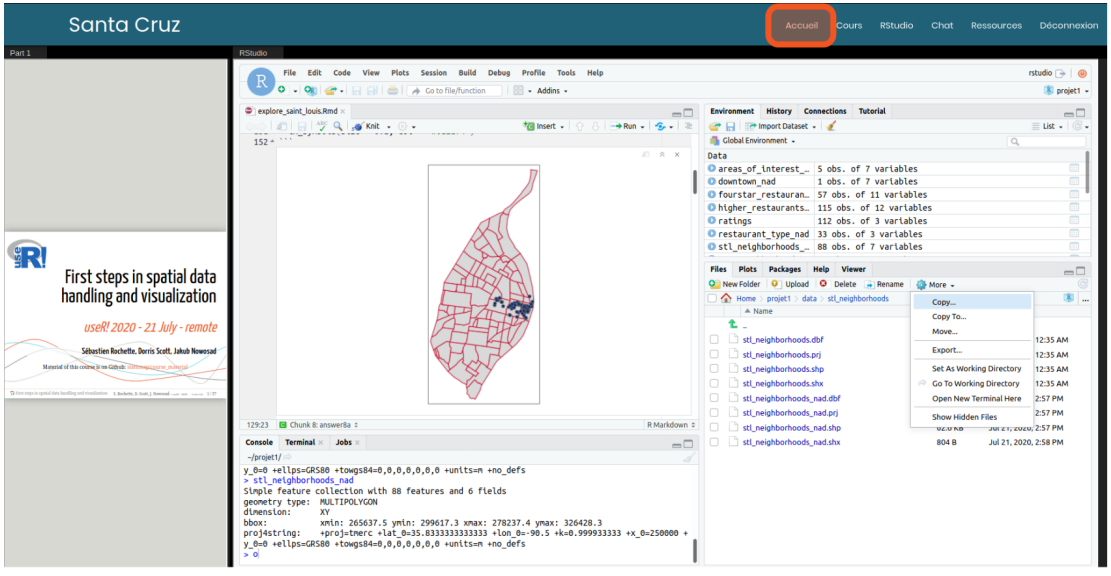
» Open the "More" Menu

» Choose "Export"

» Download

We recommend to download your full project at the end of the course.

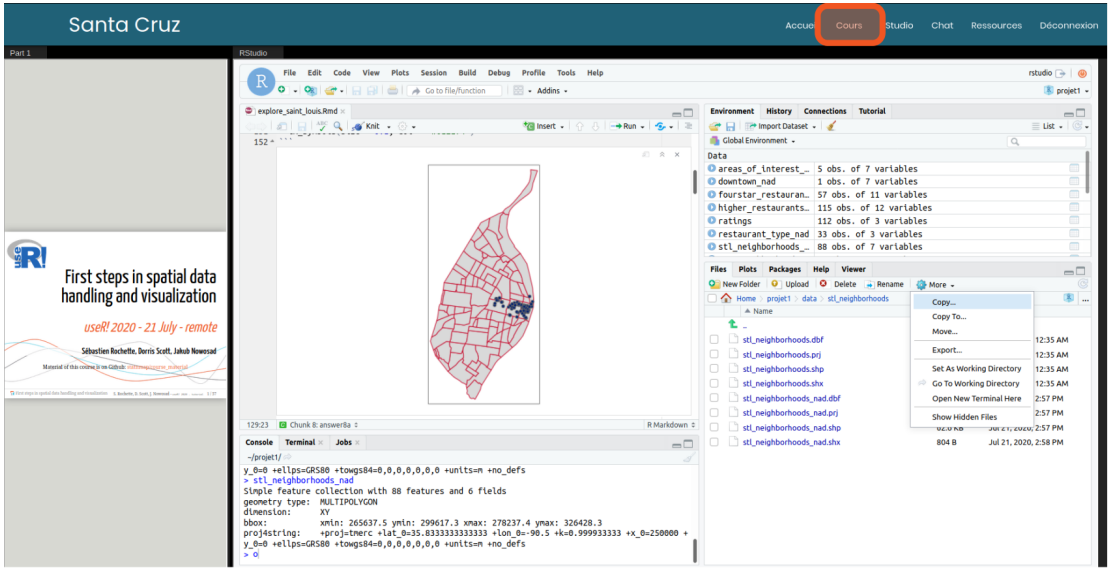
Top menu



Home

Go back to home page ("Accueil")

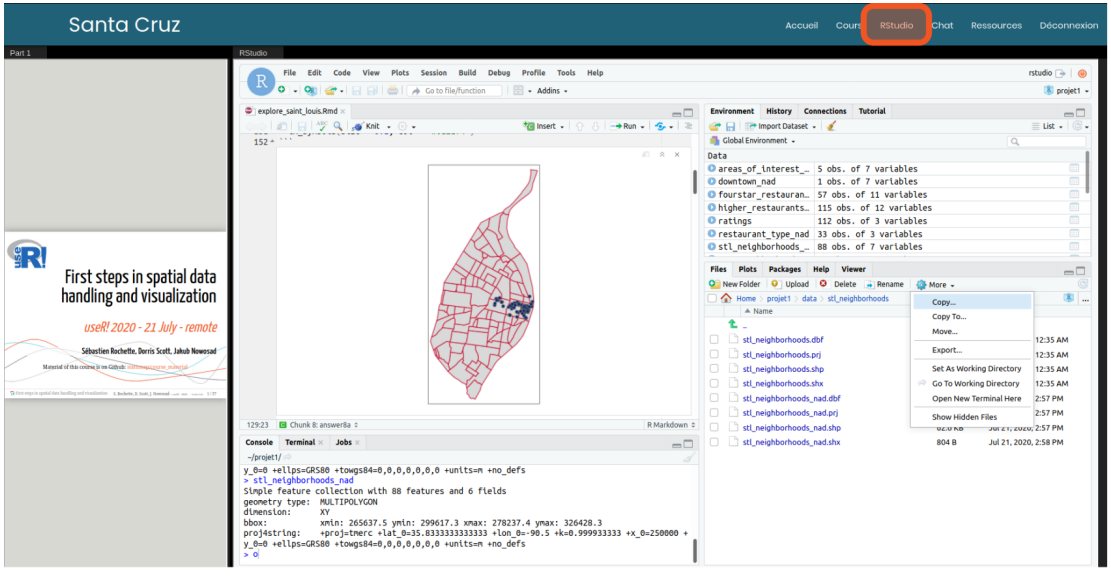
Top menu



Courses

Navigate through the list of courses available

Top menu

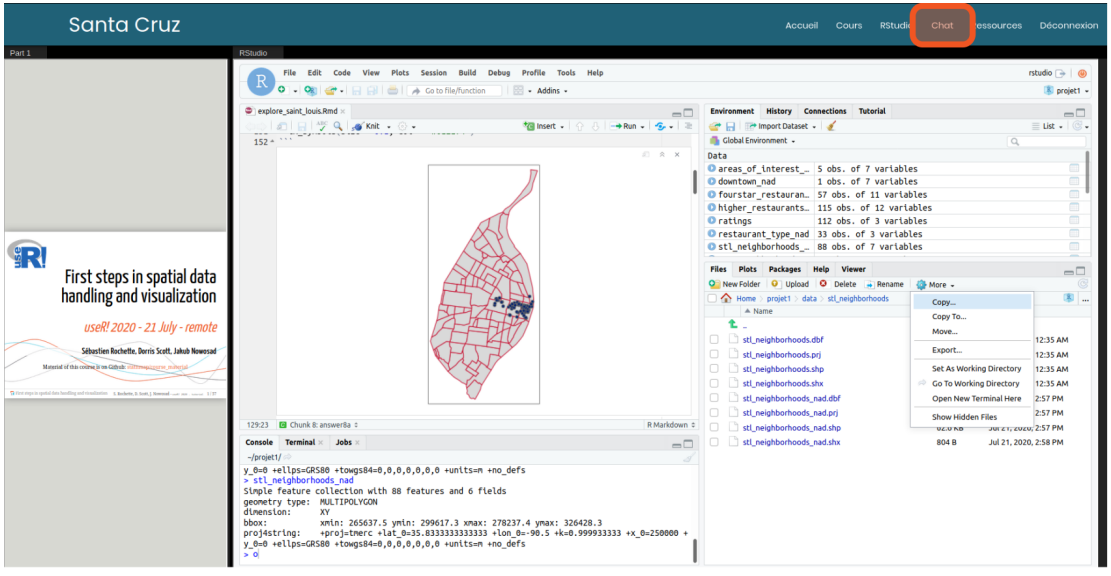


RStudio

Opens your current Rstudio session as full screen in a new tab

Note that this will close the Rstudio session in the SantaCruz interface

Top menu



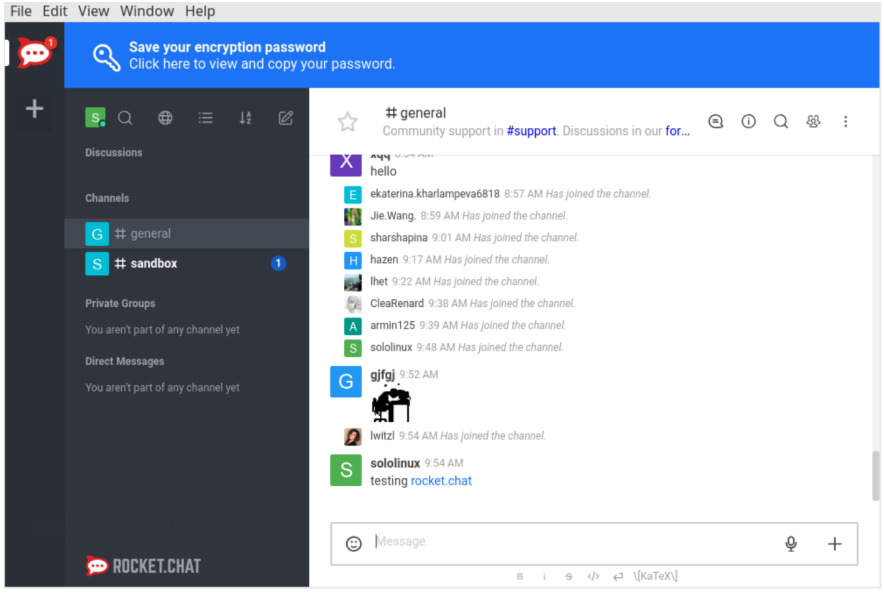
Chat

Opens a new tab with the chat interface of this course.

» Click this button

» Connect to the chat room with your credentials

Chat with us!



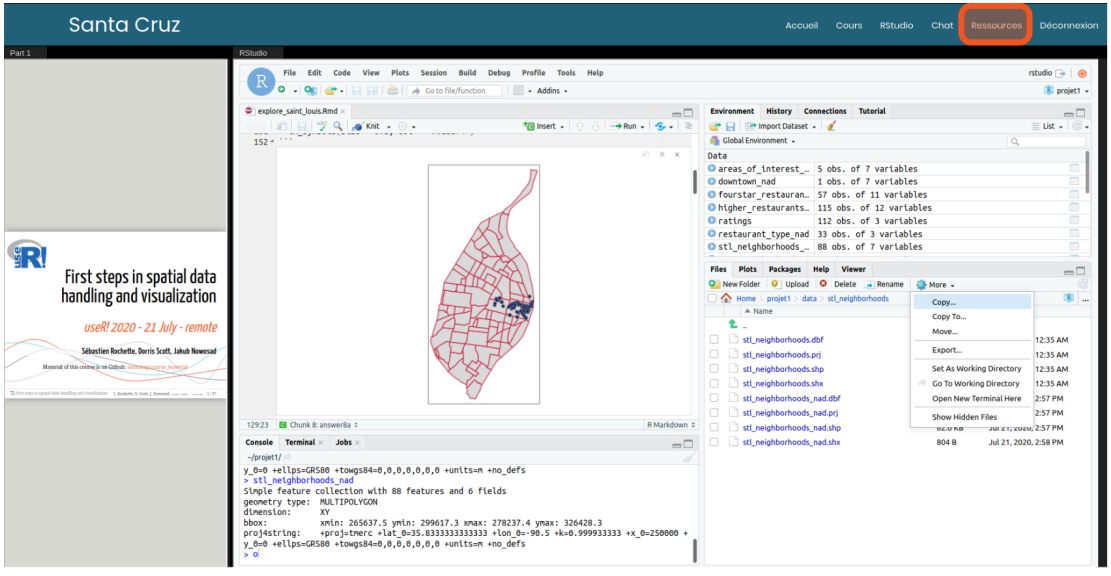
Rocket Chat

» Say "Hello" in the #general room
» Open a new thread from your message to answer "Hello to myself"

A thread allow to continue a discussion without bothering other attendees

This will be the place to discuss and share with instructors and other attendees.

Top menu



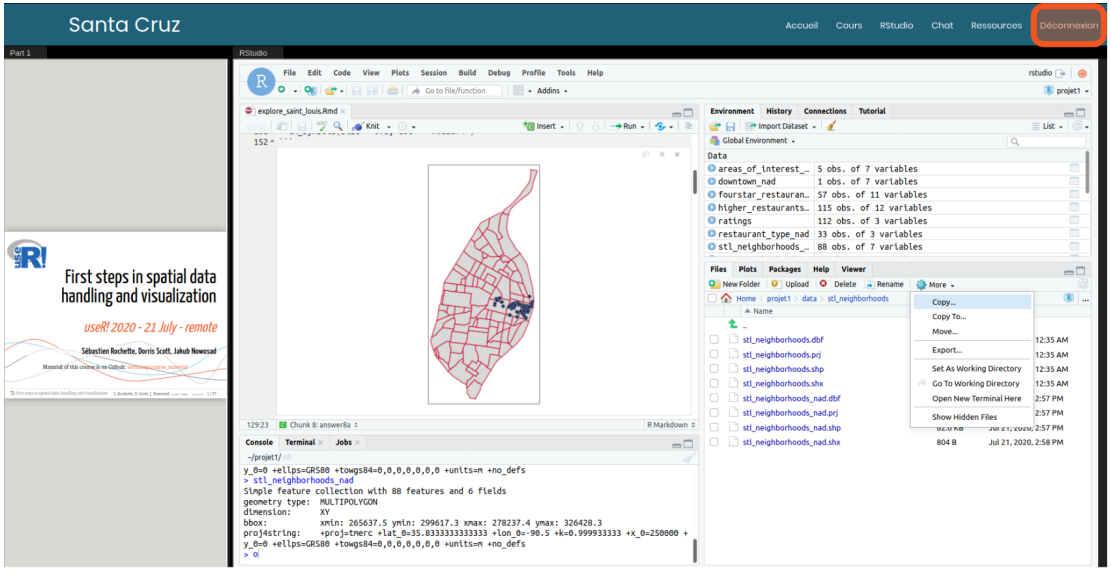
Resources

Opens a new tab with R cheatsheets as PDF

» Click this button

» Open the {sf} cheatsheet

Top menu

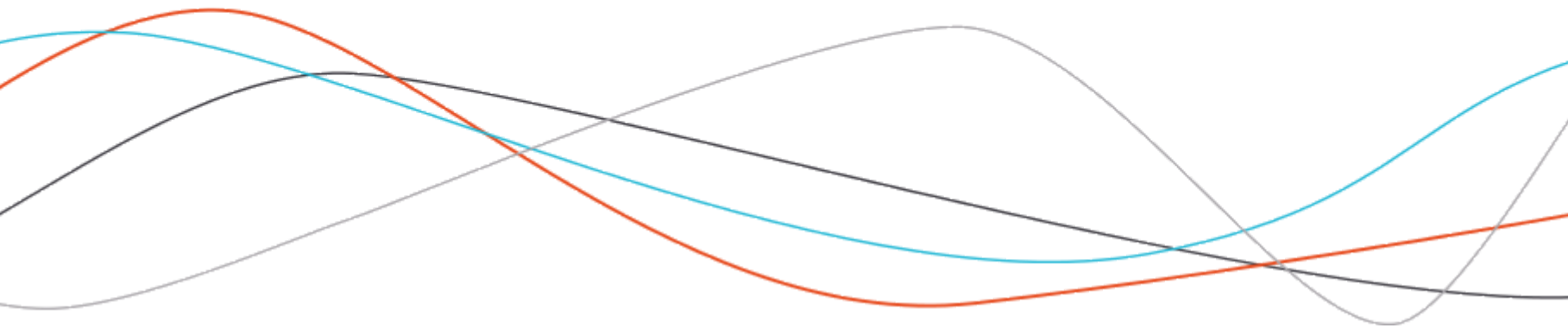


Disconnect

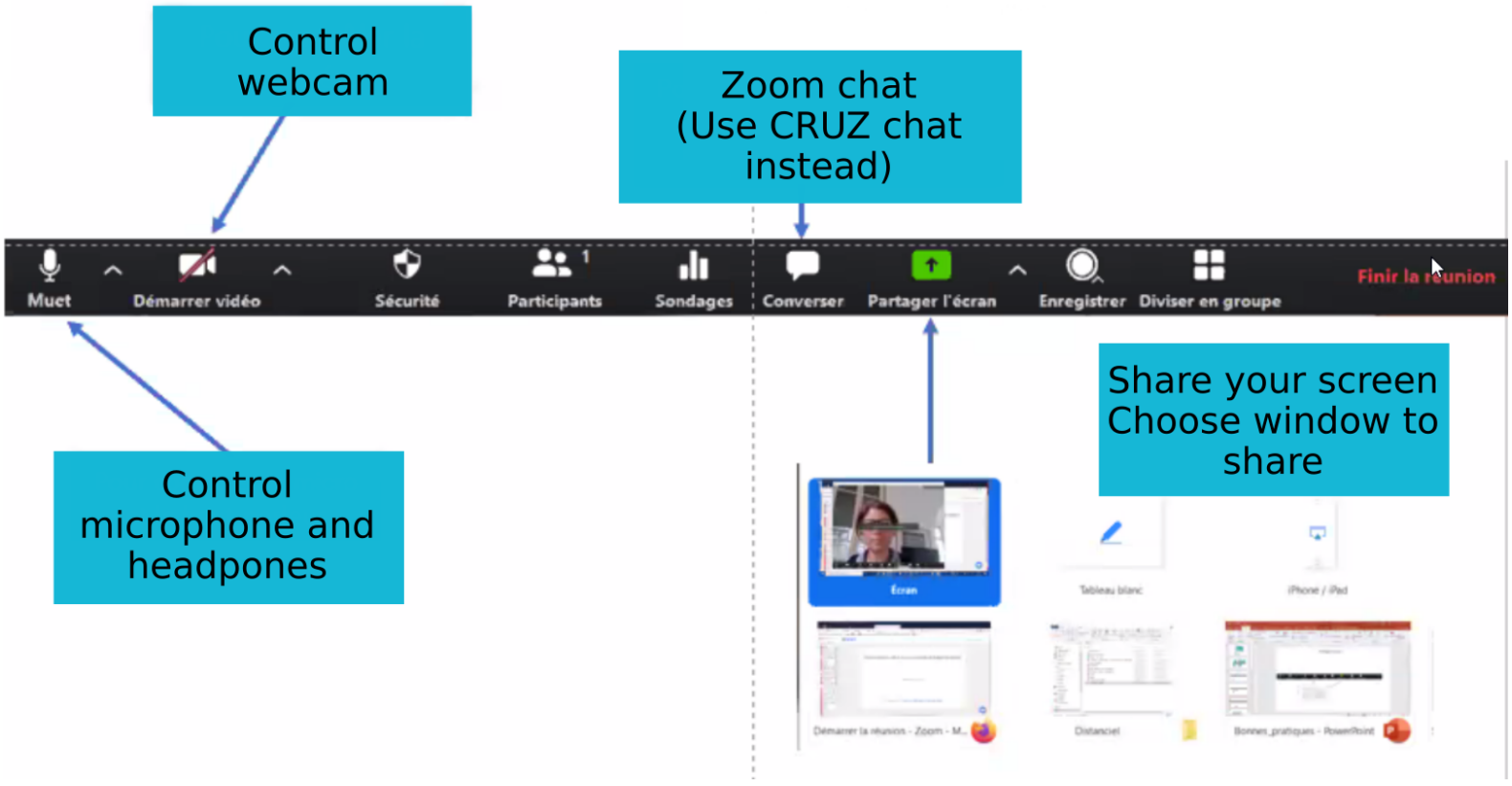
Disconnect from CRUZ

How to interact with Zoom ?

Features



Using Zoom

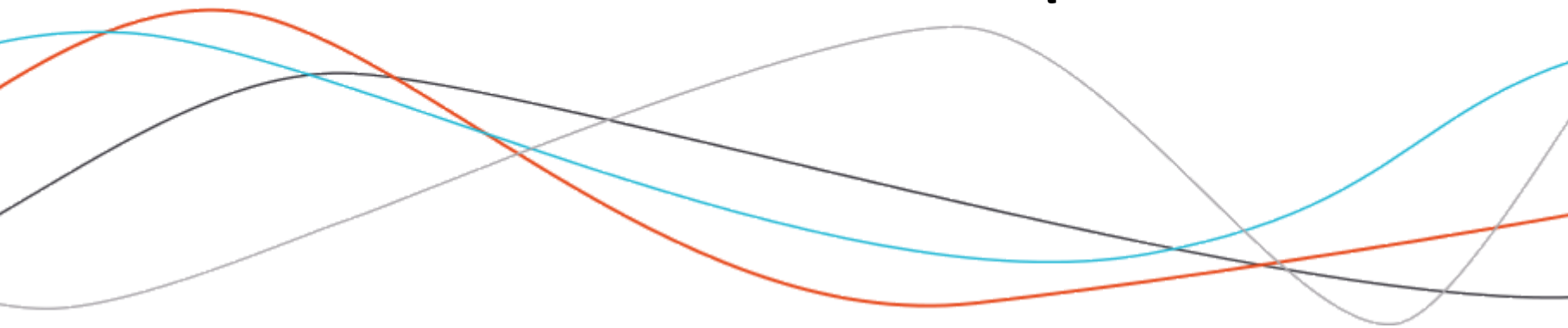

Control webcam
Zoom chat (Use CRUZ chat instead)
Muet
Démarrer vidéo
Sécurité
Participants
Sondages
Converser
Partager l'écran
Enregistrer
Diviser en groupe
Finir la réunion
Control microphone and headpones
Share your screen Choose window to share

How to interact?

- Please mute your microphone during the slides presentations
- You can share your screen during exercises if you are stuck so that we can help you
- Please cut your webcam off to reduce bandwidth use
  - Please turn it back on for interaction with trainers.
  - To avoid the impression of talking to a wall, trainers can ask 2-3 people to keep their webcam on.
- In case you feel blocked in full screen mode
  - Double click on the full screen should make it come back in a small window.
  - If not, press "Esc"
- In case of "loss" of the bar to switch off your micro/webcam
  - In full screen mode, it appears at the top of the screen
  - Move the mouse over the small green banner with the name of your computer or your name, the menu should appear.
- Tip
  - Hold the space key to temporarily open the microphone.

Drawing maps with R

{tmap} and vectors



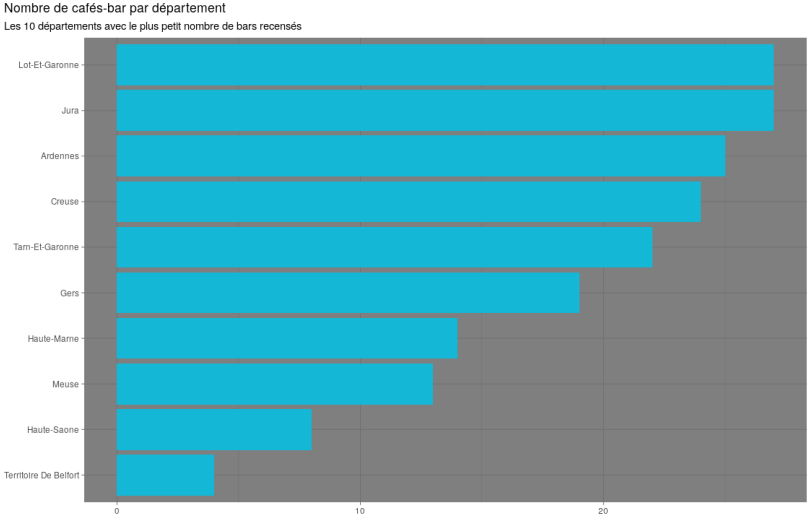
Drawing spatial data

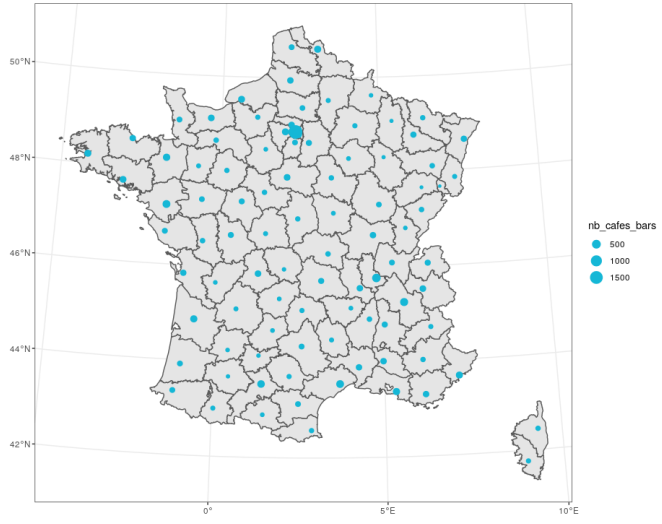
- Packages not directly dedicated for making maps
  - static : {r-base}, {ggplot2}, {rgl}
  - interactive : {plotly}

- Packages dedicated for making maps
  - static : {tmap}, {ggspatial}, {cartography}
  - interactive : {mapview}, {leaflet}, {mapdeck}

Do you really want to draw a map?

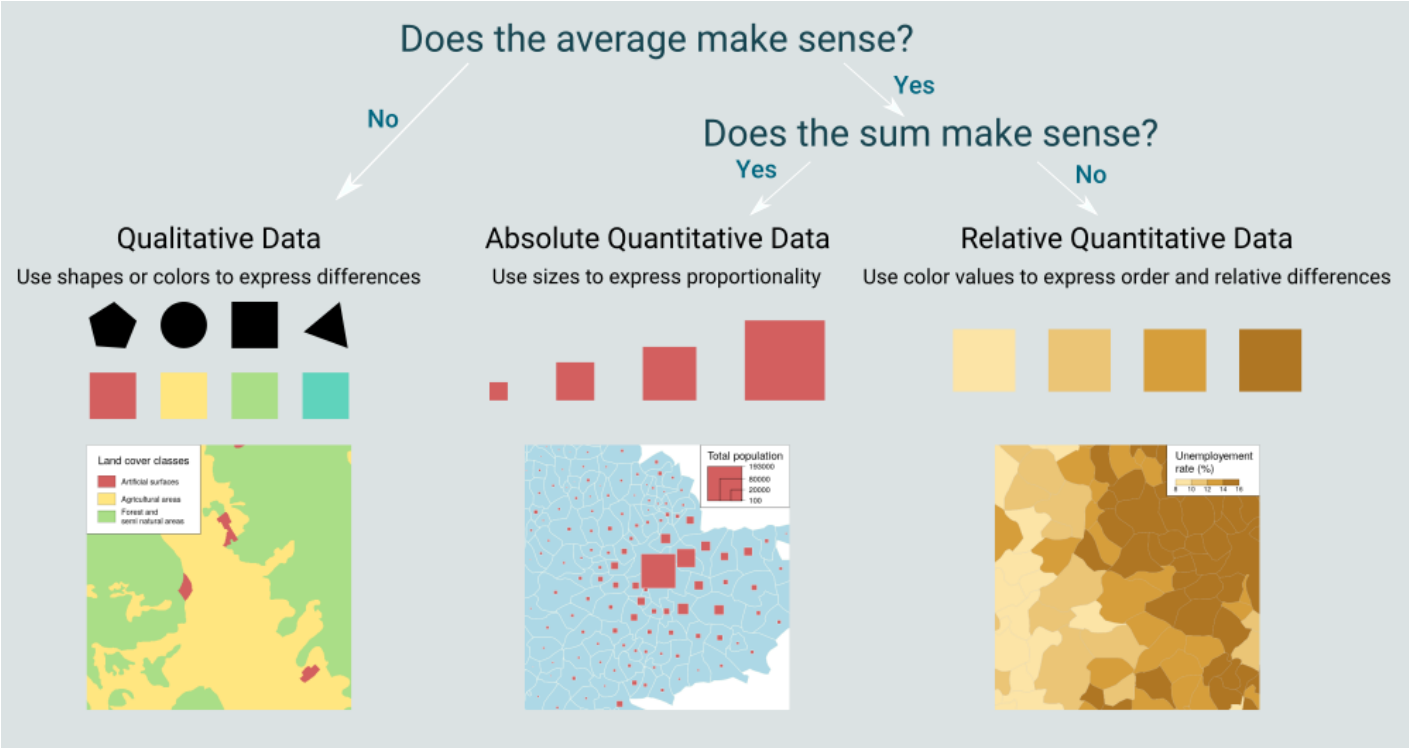
- What is your question?
- What message do you want to communicate?
- Can the map deliver this message in 3 seconds?
- **Only use maps if the spatial distribution (pattern) is meaningful**





What is the department with the smallest number of cafés-bars in France?

Visualisation rules



https://comeetie.github.io/satRday/lecture/lecture.html#maps-with-r

> If you point a to particular place in a polygon, does its representation correspond to what is happening at that particular place? (e.g. forest, number of inhabitants, density)

Read some spatial data

- `read_sf()` from `{sf}` can read shapefiles (`.shp`) and many more file spatial vector formats

```
library(sf)
# Read included dataset
# Use the tabulation for auto-completion of path
europe <- read_sf("data/europe/europe.gpkg")
departements_l93 <- read_sf("data/departements/departement.shp")
cafes_bars_l93 <- read_sf("data/cafes_bars/cafes_bars.shp")
```

{tmap} for thematic maps

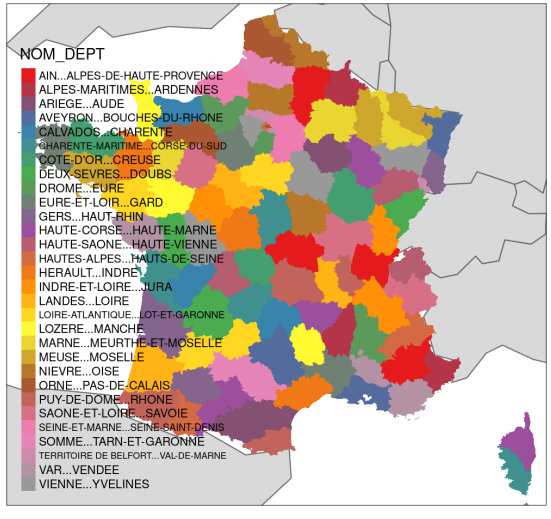
- Which dataset do you want to map?
  - `tm_shape()`
- How do you want it to be represented?
  - `tm_symbols()` : points
  - `tm_lines()` : lines
  - `tm_polygons()`, `tm_fill()`, `tm_borders()` : polygons
  - `tm_text()` : text

```
tm_shape(europe) +
  tm_polygons()
```

{tmap} for thematic maps

- Which dataset do you want to map?
  - `tm_shape()`
- How do you want it to be represented?
  - `tm_symbols()` : points
  - `tm_lines()` : lines
  - `tm_polygons()`, `tm_fill()`, `tm_borders()` : polygons
  - `tm_text()` : text

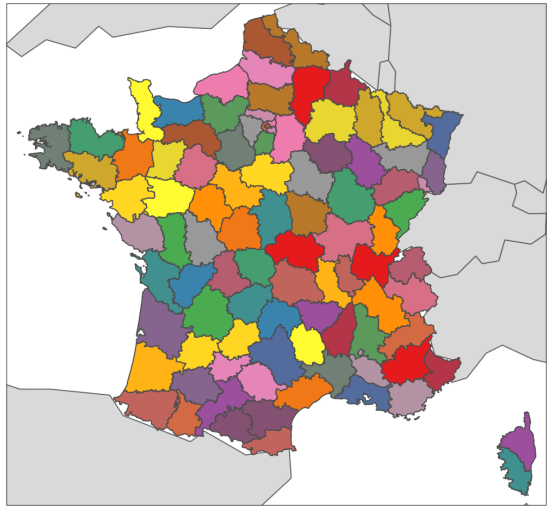
```
tm_shape(europe) +
  tm_polygons() +
tm_shape(departements_l93,
         is.master = TRUE) +
  tm_fill(col = "NOM_DEPT",
          palette = "Set1")
```



{tmap} for thematic maps

- Which dataset do you want to map?
  - `tm_shape()`
- How do you want it to be represented?
  - `tm_symbols()` : points
  - `tm_lines()` : lines
  - `tm_polygons()`, `tm_fill()`, `tm_borders()` : polygons
  - `tm_text()` : text

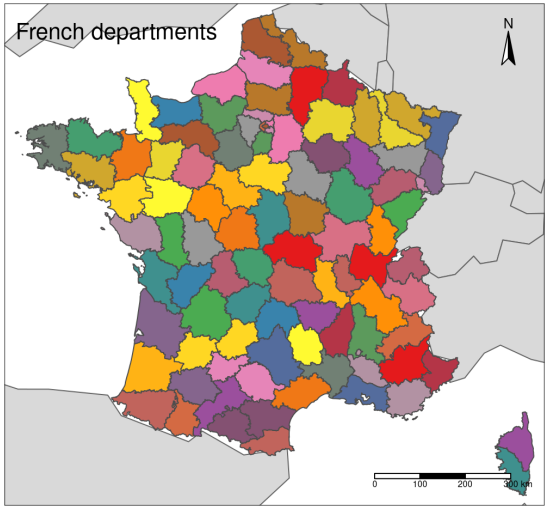
```
tm_shape(europe) +
  tm_polygons() +
tm_shape(departements_l93,
         is.master = TRUE) +
  tm_fill(col = "NOM_DEPT",
          palette = "Set1",
          legend.show = FALSE) +
  tm_borders("grey30")
```



{tmap} for thematic maps

- Add some position information
  - `tm_scale_bar()`, `tm_compass()`
- What is the general appearance of the graph?
  - `tm_layout()`

```
tm_shape(europe) +
  tm_polygons() +
tm_shape(departements_l93,
         is.master = TRUE) +
  tm_fill(col = "NOM_DEPT",
          legend.show = FALSE,
          palette = "Set1") +
  tm_borders("grey30") +
tm_scale_bar() +
tm_compass(position = c("right", "top"))
+
tm_layout(title = "French departments")
```



Quizz: What is the correct code to draw `departements_l93` map?

- `departements_l93` is a spatial object of polygons

- A

```
tm_shape(departements_l93) +
tm_symbols(col = "red") +
tm_layout(title = "France")
```

- B

```
tm_polygons(departements_l93) +
tm_layout(title = "France")
```

- C

```
tm_shape(departements_l93) +
tm_polygons() +
tm_layout(title = "France")
```

- D

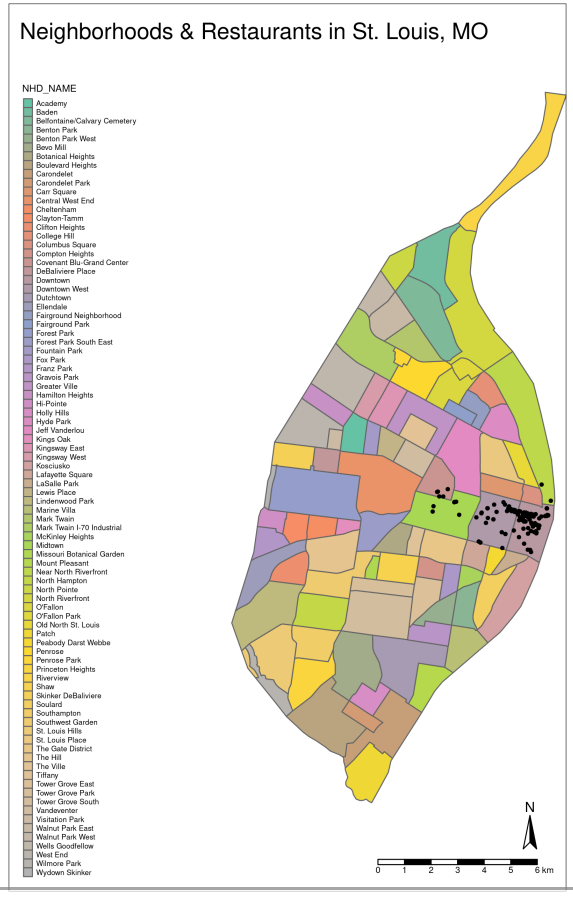
```
tm_shape(departements_l93) +
tm_polygons() +
labs(title = "France")
```

Summary

- Do you really need a map?
- Choose the correct spatial representation
- Read a shapefile with {sf}: `read_sf()`
- Draw with {tmap}:
  - `tm_shape()`: Layer to draw
  - `tm_symbols()`/`tm_lines()`/`tm_polygons()`/`tm_text()`: How to draw
  - `tm_scale_bar()`, `tm_compass()`: position information
  - `tm_layout()`: general appearance

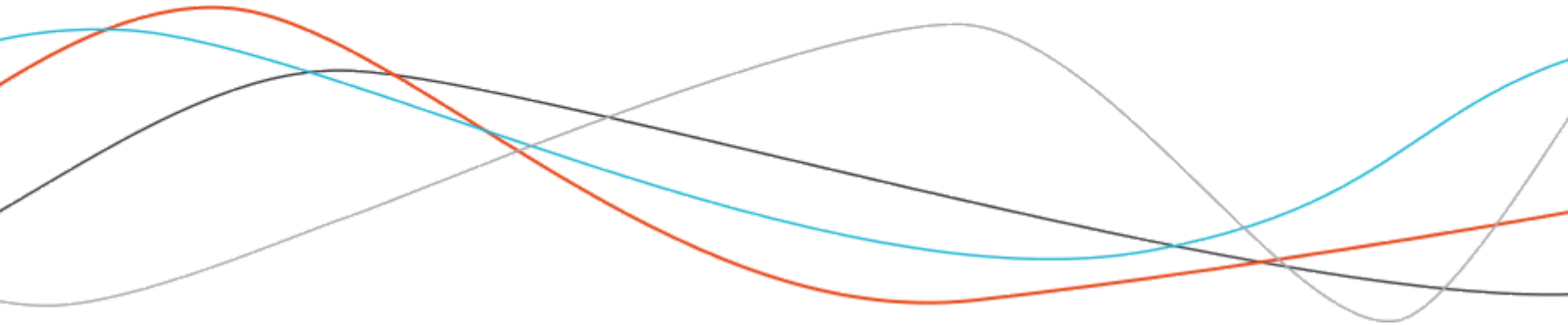
Your turn!

Exercises in: *"exo_explore_saint_louis.Rmd"*, from the **beginning** to line with title `# **Exercises for Manipulating vector data presentation**` *excluded*.



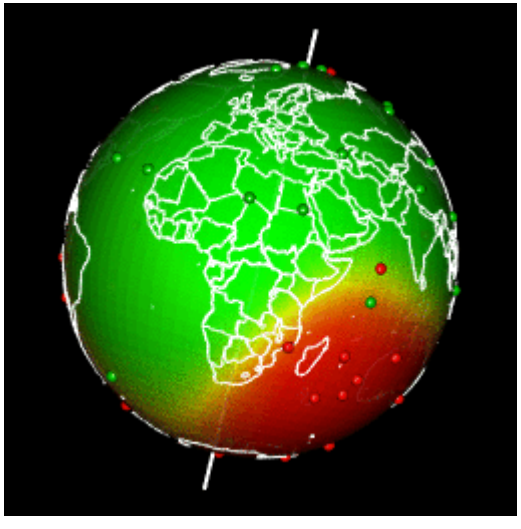
Basis of cartography

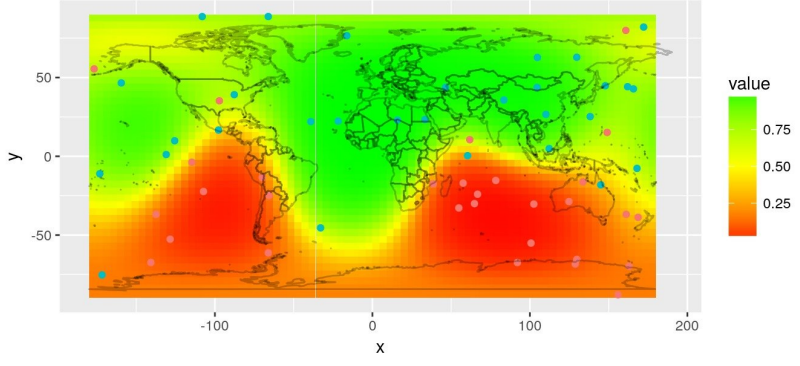
Earth is not flat...



What is a map ?

A 2-dimensional representation of all or part of our planet

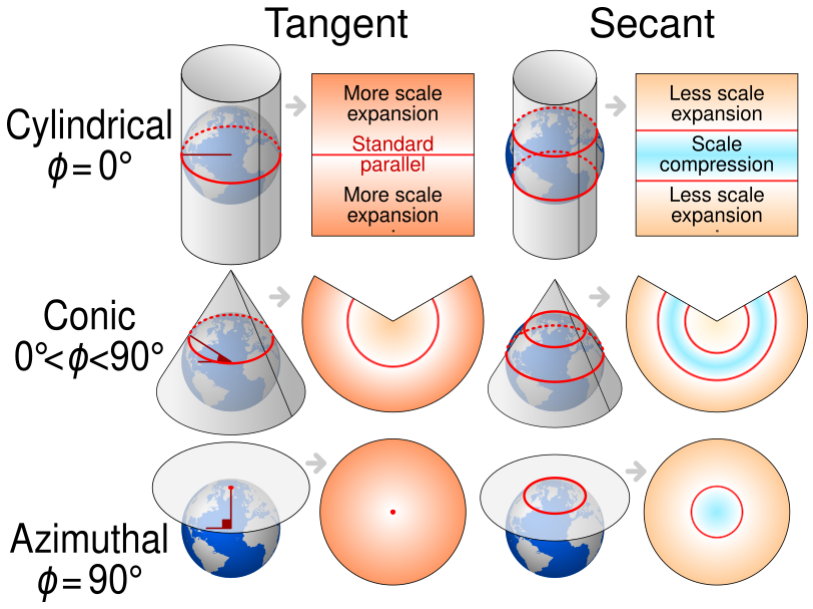




Sorry no...

What is a projection?

The source of most of your problems in cartography...



By cmglee, US government, Clindberg, Palosirkka - Globe Atlantic.svg, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=84845850

What is a projection?

- In R, you will talk about `CRS` (Coordinates Reference System)
- You need to know the projection of your datasets

> Use the name of the CRS in the name of your spatial object! This will prevent you from making many mistakes...

```
france_l93
france_wgs84
world_wintri
world_eck
```

Earth is moving...

- ...Coordinate reference systems is moving too
  - WKT (Well-Known Text)
  - Plate tectonics

sf version <0.9:

- Coordinate reference system is represented by a list with two components: `epsg` and `proj4string`

```
library(spData)
library(sf)
st_crs(nz)
```

```
## Coordinate Reference System:
## EPSG: 2193
## proj4string: "+proj=tmerc
##     +lat_0=0 +lon_0=173
##     +k=0.9996 +x_0=1600000
##     +y_0=10000000 +ellps=GRS80
##     +towgs84=0,0,0,0,0,0,0
##     +units=m +no_defs"
```

sf version >0.9 (released 2020-03-24):

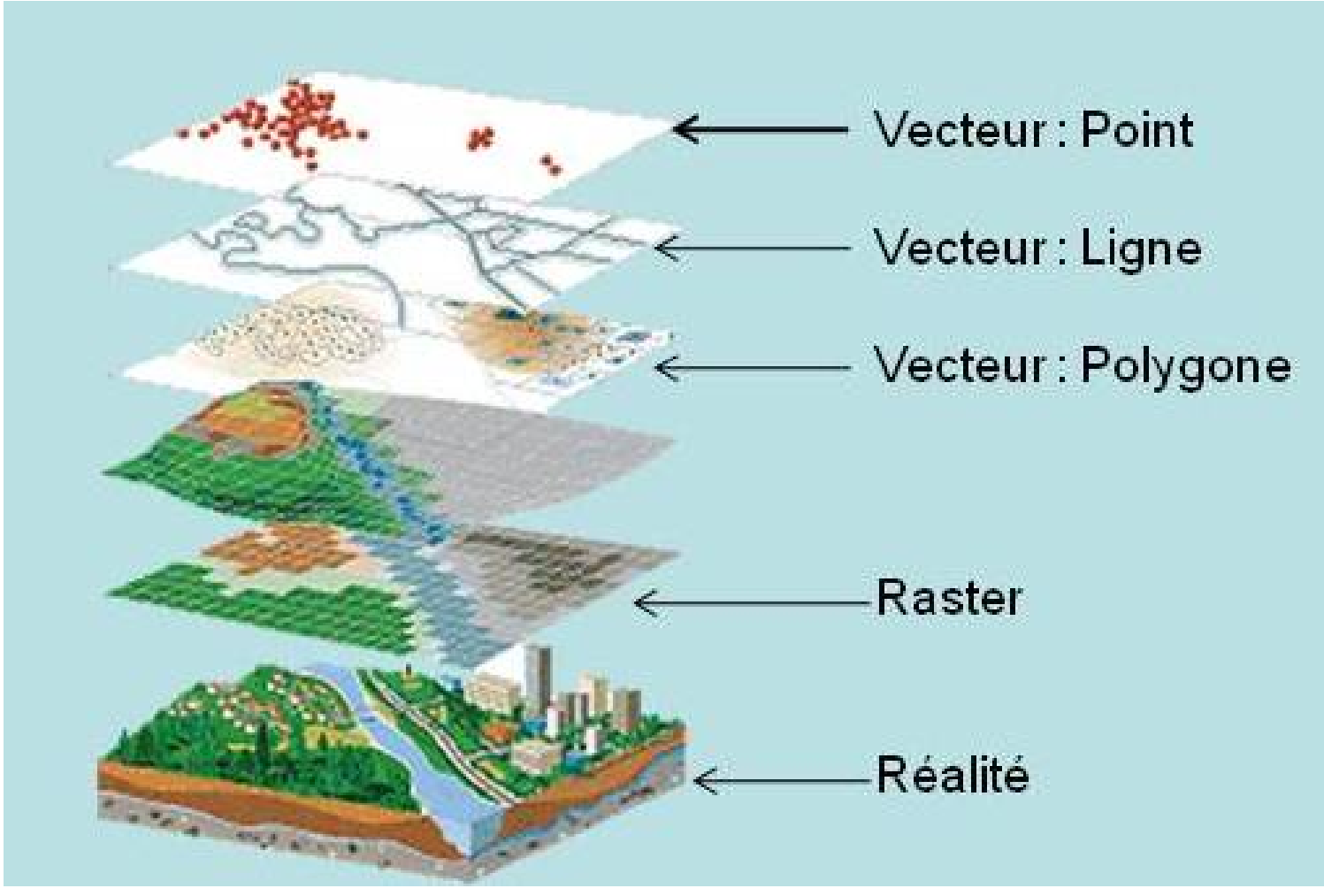
- Coordinate reference system is represented by a lists with two components, `input` and `wkt`

```
library(spData)
library(sf)
st_crs(nz)
```

```
## Coordinate Reference System:
##   User input: EPSG:2193
##   wkt:
## PROJCS["NZGD2000 / New Zealand Transverse Mercator 2000",
##     GEOGCS["NZGD2000",
##         DATUM["New_Zealand_Geodetic_Datum_2000",
##             SPHEROID["GRS 1980",6378137,298.257222101,
##                 AUTHORITY["EPSG","7019"]],
##             TOWGS84[0,0,0,0,0,0,0],
##             AUTHORITY["EPSG","6167"]],
##         PRIMEM["Greenwich",0,
##             AUTHORITY["EPSG","8901"]],
##         UNIT["degree",0.0174532925199433,
```

> J. Nowosad, R. Lovelace. https://nowosad.github.io/whyr_webinar004/

Geographic objects (entities)



Packages for reading geographic data in R

- Past: {sp} + {rgdal} + {rgeos} + {raster}
- Current packages:
  - {sf} : reading and manipulation of vector type data
    - {tidyverse} style
  - {stars} : reading and manipulation of space-time raster data
    - {tidyverse} style
  - {terra} : reading, manipulation and processing of rasters
    - Improved version of {raster}

Quizz: What is the first question to ask yourself when you have a problem with your spatial data manipulations?

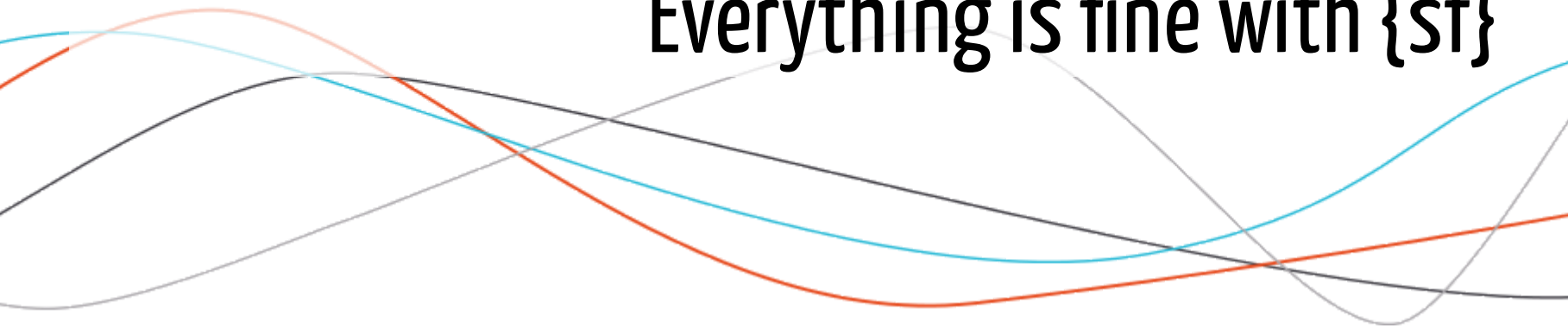
- A: Did I reboot my computer?
- B: What is the projection?
- C: What is the format of my spatial data?
- D: Who is this stupid instructor who teached me spatial data manipulation at useR! 2020?

Summary

- Projections: the main problem to deal with in cartography
- Projections: different depending on spatial extent of datasets
- Vectors data: points coordinates with attributes table
  - Lines: points ordered and linked by a line
  - Polygons: points ordered and linked by a line that closes on itself
- Rasters data: regular grid of data

Spatial vectors data: read & project

Everything is fine with {sf}



{sf} : Read geographical datasets

Read with `read_sf()`:

- All formats handled by GDAL (http://www.gdal.org/)
  - CSV, Mapinfo, Google Earth, GeoJSON, PostGIS, ...
- ESRI shapefile
  - Among most used
  - Minimum of **four** files (shp, shx, dbf, prj)
- GeoPackage
  - New format standard
  - To be prefered

```
# Map of France departments
departements_l93 <- read_sf("data/departements/departement.shp")
# Limits of a study area
study_area_l93 <- read_sf("data/departements/area_l93.gpkg")
```

Projections

- Get coordinates reference system: `st_crs()`

```
st_crs(departements_l93)
```

```
#> Coordinate Reference System:
#>   User input: RGF93_Lambert_93
#>   wkt:
#> PROJCRS["RGF93_Lambert_93",
#>     BASEGEOGCRS["RGF93",
#>         DATUM["Reseau Geodesique Francais 1993",
#>             ELLIPSOID["GRS 1980",6378137,298.257222101,
#>                 LENGTHUNIT["metre",1]],
#>             ID["EPSG",6171]],
#>         PRIMEM["Greenwich",0,
#>             ANGLEUNIT["Degree",0.0174532925199433]]],
#>     CONVERSION["unnamed",
#>         METHOD["Lambert Conic Conformal (2SP)",
#>             ID["EPSG",9802]],
#>         PARAMETER["Latitude of false origin",46.5,
#>             ANGLEUNIT["Degree",0.0174532925199433],
#>             ID["EPSG",8821]],
#>         PARAMETER["Longitude of false origin",3,
#>             ANGLEUNIT["Degree",0.0174532925199433]
```

Projections

- Project layer into new coordinates reference system: `st_transform()`

```
# Transform into geographic coordinates system
departements_wgs84 <- departements_l93 %>%
  st_transform(crs = 4326) # EPSG = 4326

st_crs(departements_wgs84)
```

```
#> Coordinate Reference System:
#>   User input: EPSG:4326
#>   wkt:
#> GEOGCRS["WGS 84",
#>     DATUM["World Geodetic System 1984",
#>         ELLIPSOID["WGS 84",6378137,298.257223563,
#>             LENGTHUNIT["metre",1]]],
#>     PRIMEM["Greenwich",0,
#>         ANGLEUNIT["degree",0.0174532925199433]],
#>     CS[ellipsoidal,2],
#>         AXIS["geodetic latitude (Lat)",north,
#>             ORDER[1],
#>             ANGLEUNIT["degree",0.0174532925199433]],
#>         AXIS["geodetic longitude (Lon)",east,
#>             ORDER[2],
```

Read text file with coordinates

- Read a csv, txt, xls, ... with the adapted package
  - {readr}, {readxl}, ...

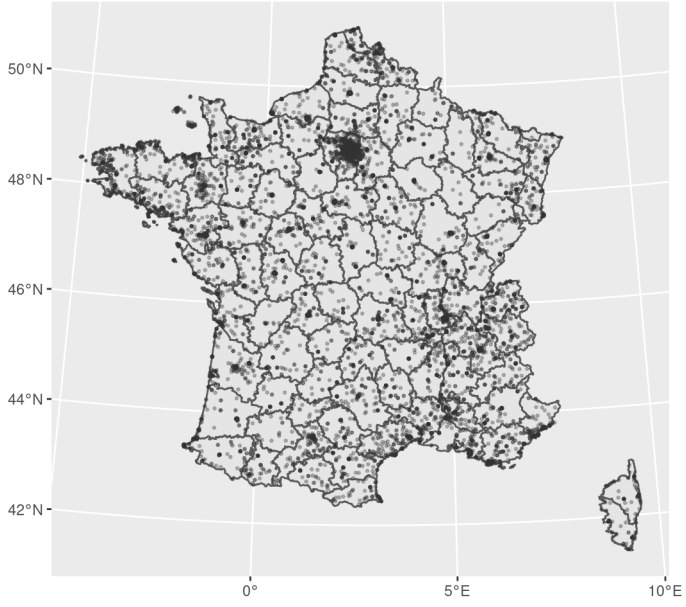
```
cafes_bars <- read_csv("data/cafes_bars.csv")
cafes_bars
```

```
#> # A tibble: 13,467 x 6
#>    amenity timestamp           user                  id    lng   lat
#>    <chr>   <dttm>              <chr>              <dbl>  <dbl> <dbl>
#>  1 bar     2014-05-25 15:41:23 Xspirt56        2880087248  0.416  46.8
#>  2 bar     2011-06-05 21:55:41 ndecaris        1313547128 -0.466  46.3
#>  3 bar     2011-02-22 14:19:50 Syl             1166456913  5.71   45.2
#>  4 bar     2013-09-16 12:42:04 Thibaut75011    2460347252  2.37   48.9
#>  5 bar     2012-11-17 05:48:19 Agibi           2017656504  0.129  49.5
#>  6 bar     2015-01-12 19:19:18 woodpeck_repair 2273314756  4.84   45.8
#>  7 bar     2013-08-02 20:51:21 Olivier82       2405180491  1.48   44.2
#>  8 bar     2013-05-08 16:04:23 Benat Iza       2296077080 -1.79   43.4
#>  9 bar     2015-05-09 17:49:03 Weslape         3507311321  7.76   48.6
#> 10 bar     2012-08-18 12:49:59 lttnono         1871307487 -4.07   48.2
#> # … with 13,457 more rows
```

Read text file with coordinates

- Use `st_as_sf()` to transform into spatial dataset

```
cafes_bars_wgs84 <- cafes_bars %>%
  st_as_sf(coords = c("lng", "lat"), crs = 4326)
```



Write spatial dataset on disk

Write with `write_sf()` :

- All formats handled by GDAL (http://www.gdal.org/)
  - CSV, Mapinfo, Google Earth, GeoJSON, PostGIS, Shp, ...
- GPKG format (GeoPackage)
  - New standard of OGC
  - Recommended

```
write_sf(point_wgs84, "new-directory/point_wgs84.gpkg")
```

Quizz: What is the correct way to deal with the following text file with coordinates to be used as a spatial points dataset?

- Let `"data_coords.txt"` be a text file at the root of your project
- Position of points was taken using a GPS in France
- Note: `2154`: Lambert 93, France, projected; `4326`: Geographical coordinates

```
#>    lat lng sample
#> 1 48.5 2.0      A
#> 2 49.0 2.5      B
#> 3 47.0 3.0      C
```

- A

```
d_text <- read_txt("data_coords.txt")
pts_wgs84 <- d_text %>%
  st_as_sf(coords = c("lng", "lat"),
           crs = 4326)
```

- B

```
d_text <- read_txt("data_coords.txt")
pts_wgs84 <- d_text %>%
  st_transform(crs = 4326)
```

- C

```
d_text <- read_txt("data_coords.txt")
pts_l93 <- d_text %>%
  st_as_sf(coords = c("lat", "lng"),
           crs = 2154)
```

- D

```
pts_wgs84 <-
read_sf("data_coords.txt")
```

Summary

- Package : {sf}
- Read : `read_sf()`
- Write : `write_sf()`
- Get projection : `st_crs()`
- Transform, Project : `st_transform()`
- Text file into spatial : `st_as_sf()`

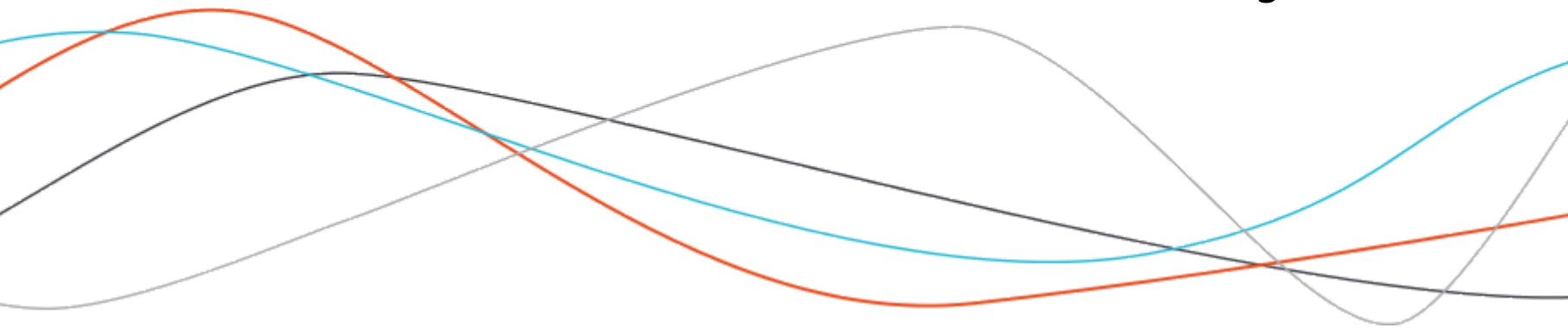
Your turn!

Exercises in: *"exo_explore_saint_louis.Rmd"*

- Start from title `# **Exercises for Spatial vectors data: read & project**`
- Stop on line with title `# **Exercises for Manipulating vector data: Use the {tidyverse}**` *excluded*.

Manipulating vector data

Use the {tidyverse}



Vector data format

- With {sf}, data are rectangular

```
library(sf)
# Read example data
# vector
departements_l93 <- read_sf("data/departements/departement.shp")
# Text file - number of inhabitants
dept_inhabitants <- read_csv("data/departements/repartition_dpt_clean_2015.csv")
```

```
departements_l93
```

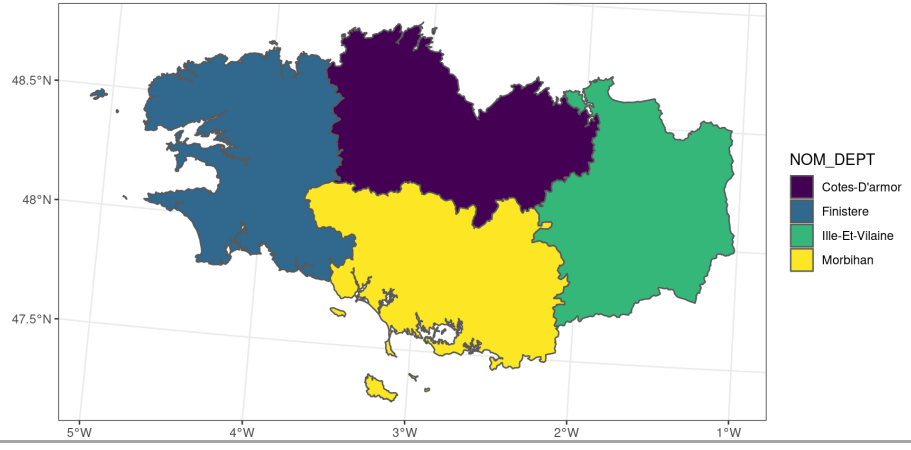
```
#> Simple feature collection with 96 features and 11 fields
#> geometry type:  MULTIPOLYGON
#> dimension:      XY
#> bbox:           xmin: 99217.1 ymin: 6049646 xmax: 1242417 ymax: 7110480
#> projected CRS:  RGF93_Lambert_93
#> # A tibble: 96 x 12
#>    ID_GEOFLA CODE_DEPT NOM_DEPT CODE_CHF NOM_CHF X_CHF_LIEU Y_CHF_LIEU
#>    <chr>     <chr>     <chr>    <chr>    <chr>        <int>      <int>
#>  1 DEPARTEM… 39        JURA     300      LONS-L…     895198    6622537
#>  2 DEPARTEM… 42        LOIRE    218      SAINT-…     808646    6482549
#>  3 DEPARTEM… 76        SEINE-M… 540      ROUEN       562355    6928721
#>  4 DEPARTEM… 89        YONNE    024      AUXERRE     742447    6744261
```

Operations on rows and columns

All that you know with {dplyr} works on {sf} objects

- `%>%`
- `select`, `mutate` for attributs (= columns)
- `filter`, `arrange` for entities (= rows)

```
bretagne_l93 <-
  departements_l93 %>%
  mutate(NOM_DEPT = str_to_title(NOM_DEPT)) %>%
  select(CODE_DEPT, NOM_DEPT, NOM_REG) %>%
  filter(NOM_REG == "BRETAGNE")
```

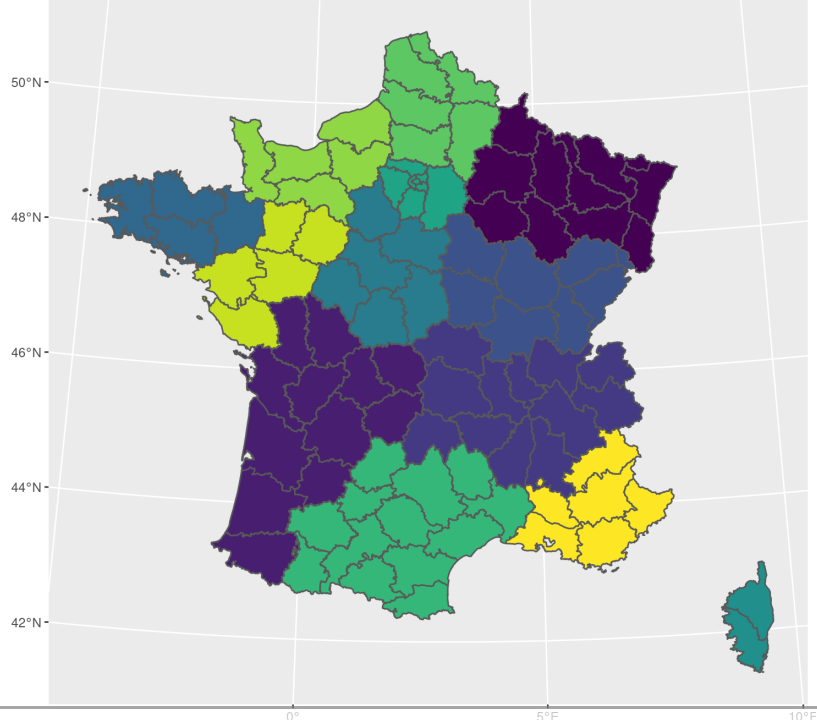


Grouped operations

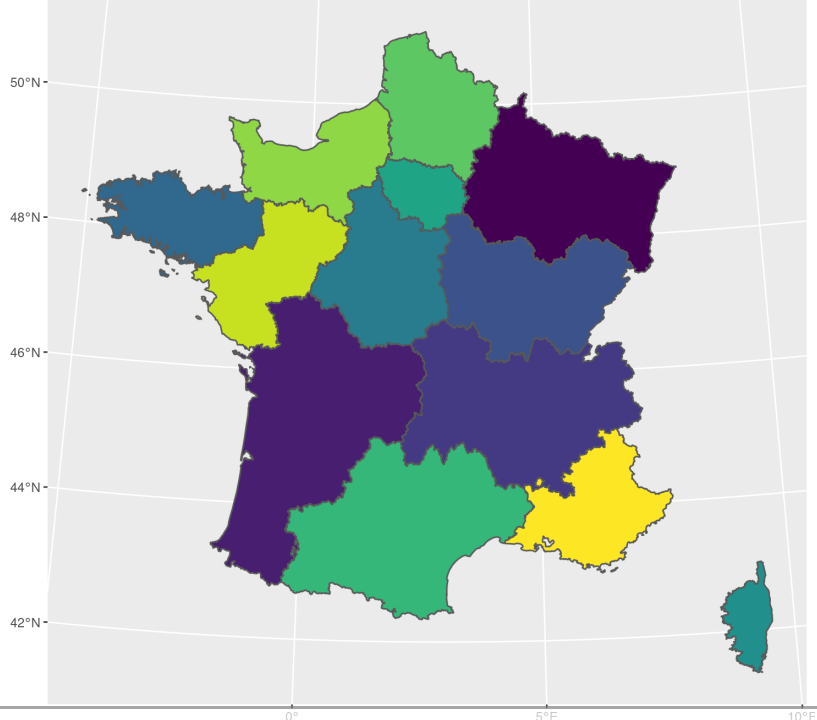
- Merge entities with `group_by` + `summarize`

```
region_l93 <- departements_l93 %>%
  group_by(NOM_REG) %>%
  summarize()
```

Departments



Regions



Join with classical datasets

- Between shapefile and dataset with `*_join`
  - `left_join()`, `inner_join()`, `full_join()`, ...
  - Read external classical dataset

```
# Text file - number of inhabitants
dept_inhabitants <- read_csv("data/departements/repartition_dpt_clean_2015.csv")
```

+ Content of shapefile:

```
departements_l93
```

```
#> Simple feature collection with 96 features and 11 fields
#> geometry type:  MULTIPOLYGON
#> dimension:      XY
#> bbox:           xmin: 99217.1 ymin: 6049646 xmax: 1242417 ymax: 7110480
#> projected CRS:  RGF93_Lambert_93
#> # A tibble: 96 x 12
#>    ID_GEOFLA CODE_DEPT NOM_DEPT CODE_CHF NOM_CHF X_CHF_LIEU Y_CHF_LIEU
#>    <chr>     <chr>     <chr>    <chr>    <chr>        <int>      <int>
#>  1 DEPARTEM… 39        JURA     300      LONS-L…     895198    6622537
#>  2 DEPARTEM… 42        LOIRE    218      SAINT-…     808646    6482549
#>  3 DEPARTEM… 76        SEINE-M… 540      ROUEN       562355    6928721
#>  4 DEPARTEM… 89        YONNE    024      AUXERRE     742447    6744261
#>  5 DEPARTEM… 68        HAUT-RH… 066      COLMAR     1024125    6784581
```

Join with classical datasets

- Between shapefile and dataset with `*_join`

- Department shapefile `departements_l93`

```
glimpse(departements_l93, width = 30)
```

```
#> Rows: 96
#> Columns: 12
#> $ ID_GEOFLA  <chr> "DEPARTE…
#> $ CODE_DEPT  <chr> "39", "4…
#> $ NOM_DEPT   <chr> "JURA", …
#> $ CODE_CHF   <chr> "300", "…
#> $ NOM_CHF    <chr> "LONS-LE…
#> $ X_CHF_LIEU <int> 895198, …
#> $ Y_CHF_LIEU <int> 6622537,…
#> $ X_CENTROID <int> 886172, …
#> $ Y_CENTROID <int> 6641548,…
#> $ CODE_REG   <chr> "27", "8…
#> $ NOM_REG    <chr> "BOURGOG…
#> $ geometry   <MULTIPOLYGON [m]> …
```

- Number of inhabitants by age by department in `dept_inhabitants`

```
glimpse(dept_inhabitants, width = 30)
```

```
#> Rows: 101
#> Columns: 7
#> $ ID_DPT            <chr> "…
#> $ X1                <chr> "…
#> $ `Age_40-59 ans`   <dbl> 1…
#> $ `Age_0-19 ans`    <dbl> 1…
#> $ `Age_75 ans et +` <dbl> 5…
#> $ `Age_20-39 ans`   <dbl> 1…
#> $ `Age_60-74 ans`   <dbl> 9…
```

Join with classical datasets

```r
# Join database with shapefile by attributes
departements_inhabitants_l93 <- departements_l93 %>%
 inner_join(dept_inhabitants, by = c("CODE_DEPT" = "ID_DPT"))

glimpse(departements_inhabitants_l93, width = 40)
```

```
#> Rows: 96
#> Columns: 18
#> $ ID_GEOFLA         <chr> "DEPARTEM00…
#> $ CODE_DEPT         <chr> "39", "42",…
#> $ NOM_DEPT          <chr> "JURA", "LO…
#> $ CODE_CHF          <chr> "300", "218…
#> $ NOM_CHF           <chr> "LONS-LE-SA…
#> $ X_CHF_LIEU        <int> 895198, 808…
#> $ Y_CHF_LIEU        <int> 6622537, 64…
#> $ X_CENTROID        <int> 886172, 795…
#> $ Y_CENTROID        <int> 6641548, 65…
#> $ CODE_REG          <chr> "27", "84",…
#> $ NOM_REG           <chr> "BOURGOGNE-…
#> $ geometry          <MULTIPOLYGON [m]> …
#> $ X1                <chr> "D39 Jura",…
#> $ `Age_40-59 ans`   <dbl> 71842, 1962…
#> $ `Age_0-19 ans`    <dbl> 61505, 1865…
```

Quizz: What code do you choose to join the external classical dataset `restaurants` with the spatial dataset `france_l93` if you want to keep spatial entities but only where there is a match?

restaurants

```
#>         department restaurants
#> 1        CALVADOS        1500
#> 2 ALPES-MARITIMES        3000
#> 3          CANTAL        2000
```

- A

```
france_l93 %>%
group_by(restaurants)
```

- B

```
france_l93 %>%
left_join(restaurants)
```

- C

```
restaurants %>%
inner_join(france_l93)
```

france_l93 (extract)

```
#> # A tibble: 96 x 2
#>   department
geometry
#>   <chr>                <MULTIPOLYGON
[m]>
#> 1 JURA       (((886244.2 6641236,
88619…
#> 2 LOIRE      (((764370.3 6544751,
76438…
#> # … with 94 more rows
```

- D

```
france_l93 %>%
inner_join(restaurants)
```

Summary

- {dplyr} grammar on attribute table
  - `select()`, `mutate()`, `filter()`, ...
  - `inner_join()`, `left_join()`, `full_join()`, `anti_join()`, ...
- {dplyr} grammar on attribute + geometry
  - `group_by()` + `summarise()`
- Note that there is also compatibility with {tidyr} functions

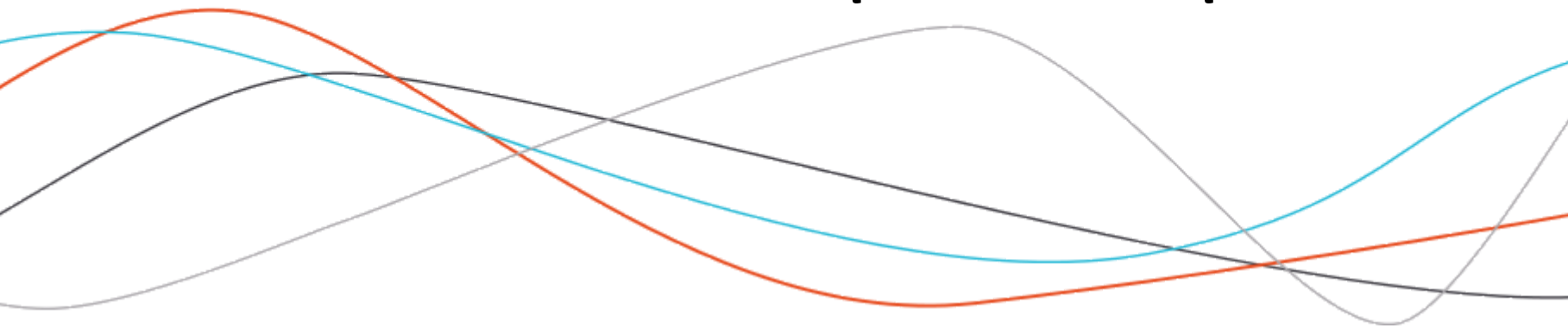
Your turn!

Exercises in: *"exo_explore_saint_louis.Rmd"*

- Start from title `# **Exercises for Manipulating vector data: Use the {tidyverse}**`
- Stop on line with title `# **Exercises for Manipulating vector data: Spatial manipulations**` *excluded*.

Manipulating vector data

Spatial manipulations



Read data used in this chapter

```
library(sf)
# Lire vos données de test
# vector
departements_l93 <- read_sf("data/departements/departement.shp")
# cafes-bars
cafes_bars_l93 <- read_sf("data/cafes_bars/cafes_bars.shp")
# Study area
area_l93 <- read_sf("data/departements/area_l93.gpkg")
```

Attributes

- `st_crs()` : Get the coordinates reference system
- `st_bbox()` : Get the spatial extent

```
departements_l93 %>% st_bbox()
```

```
#>      xmin      ymin      xmax      ymax
#>   99217.1 6049646.3 1242417.2 7110480.1
```

Spatial join

- Spatial join with `st_join`
  - *like* `left_join` *using geographical positions*
- Spatial intersection with `st_intersection`
  - *like* `inner_join` *using geographical positions*
- Intersection of points / polygons / lines with polygons

```
bretagne_l93 <- departements_l93 %>%
  filter(NOM_REG == "BRETAGNE")
```

```
cafes_bars_bretagne_l93 <-
  cafes_bars_l93 %>%
  st_intersection(bretagne_l93)
```

```
names(cafes_bars_l93)
```

```
#> [1] "amenity"   "timestamp" "user"
"id"        "geometry"
```

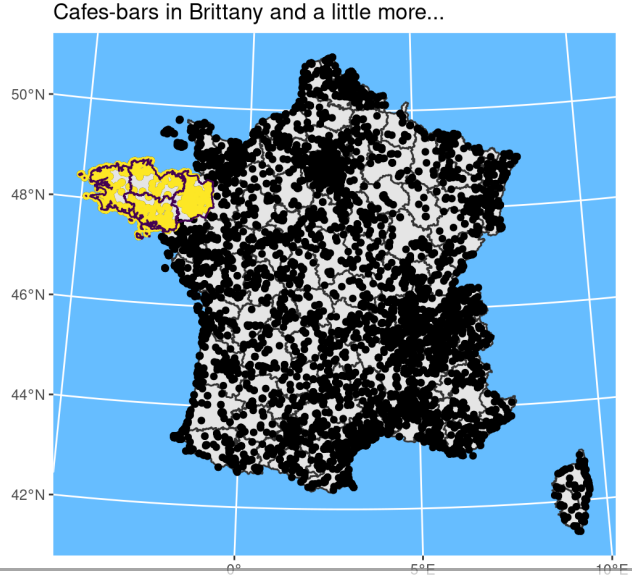
```
names(cafes_bars_bretagne_l93)
```

```
#>  [1] "amenity"    "timestamp"  "user"
"id"         "ID_GEOFLA"
#>  [6] "CODE_DEPT"  "NOM_DEPT"
"CODE_CHF"   "NOM_CHF"    "X_CHF_LIEU"
#> [11] "Y_CHF_LIEU" "X_CENTROID"
"Y_CENTROID" "CODE_REG"   "NOM_REG"
#> [16] "geometry"
```

Spatial join

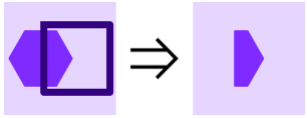
Presentation of spatial intersection using `st_intersection`

```
ggplot() +
  geom_sf(data = departements_l93, colour = "grey20") +
  geom_sf(data = cafes_bars_l93) + # all points in black
  geom_sf(data = cafes_bars_bretagne_l93, colour = "#FDE725") + # yellow points
  geom_sf(data = bretagne_l93, fill = NA, colour = "#440154") +
  coord_sf(crs = 2154, datum = 4326) +
  theme(panel.background = element_rect(fill = "#66BDFF")) +
  labs(title = "Cafes-bars in Brittany and a little more...")
```

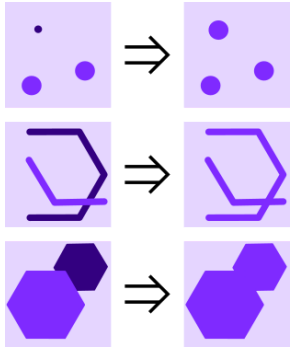


Geometric operations

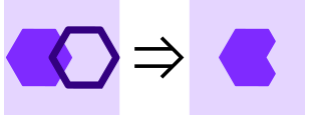
- Crop without keeping attributes with `st_crop(x, y)`



- Union with `st_union(x, y)`

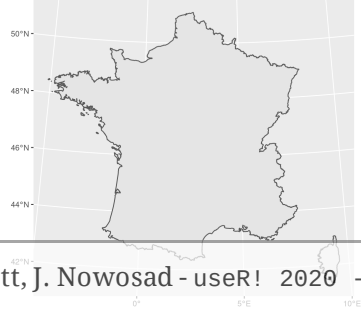


- Difference with `st_difference(x, y)`



- Union on its own entities `st_union(x)`

```
departements_l93 %>% st_union()
```



Geometric measures

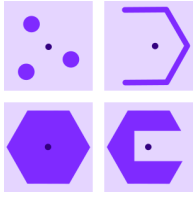
- `st_area(x)` calculate areas of polygons
- `st_distance(x, y)` calculate distances between x and y
- `st_length(x)` calculate length of a line

```
# Ajouter comme variable à la couche spatiale
departements_l93 %>%
  mutate(area = st_area(geometry)) %>%
  select(CODE_DEPT, area)
```

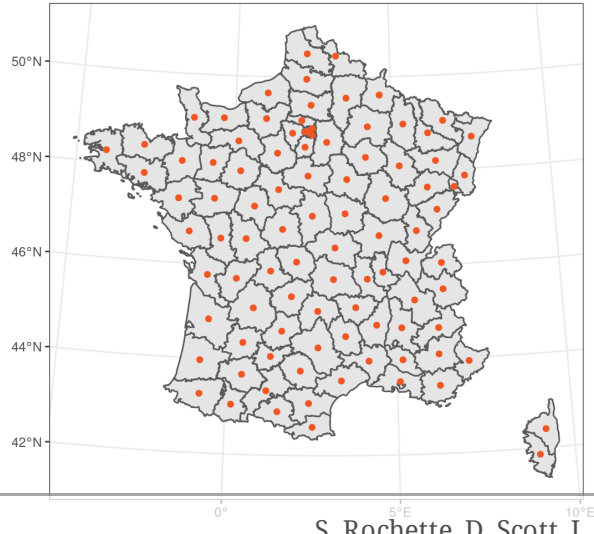
```
#> Simple feature collection with 96 features and 2 fields
#> geometry type:  MULTIPOLYGON
#> dimension:      XY
#> bbox:           xmin: 99217.1 ymin: 6049646 xmax: 1242417 ymax: 7110480
#> projected CRS:  RGF93_Lambert_93
#> # A tibble: 96 x 3
#>    CODE_DEPT       area                                              geometry
#>    <chr>          [m^2]                                   <MULTIPOLYGON [m]>
#>  1 39        5039577584 (((886244.2 6641236, 886199.4 6641257, 885970.3 6641339…
#>  2 42        4797372418 (((764370.3 6544751, 764380.4 6544766, 764395.5 6544776…
#>  3 76        6327702279 (((511688.8 6966777, 511693.9 6966792, 512131 6967074, …
#>  4 89        7451282080 (((709449.1 6765661, 709449.1 6765666, 709359.6 6765712…
#>  5 68        3528390720 (((992779.1 6768618, 992764.2 6768628, 992744.9 6768708…
```

Other geometric operations

- Get centroids of entities with `st_centroid()`

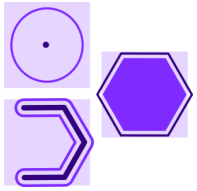


```
dept_centroid_l93 <- departements_l93 %>% st_centroid()
ggplot() +
  geom_sf(data = departements_l93) +
  geom_sf(data = dept_centroid_l93, colour = "#f15522") +
  theme_bw()
```

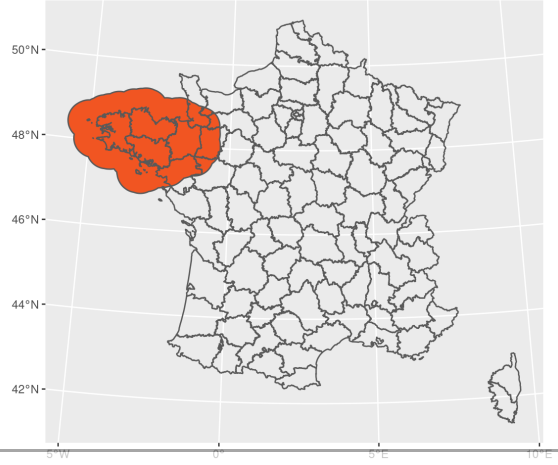


Other geometric operations

- Calculate buffer area `st_buffer()`



```
bretagne_buffer_l93 <- departements_l93 %>%
  filter(NOM_REG == "BRETAGNE") %>% st_union() %>%
  st_buffer(dist = units::set_units(50, km))
ggplot() +
  geom_sf(data = bretagne_buffer_l93, fill = "#f15522") +
  geom_sf(data = departements_l93, fill = NA)
```



Quizz: What function do you use to, at the same time, die-cut a spatial points dataset using a spatial polygons dataset, and, associate variables from the polygons to the points?

- A: `st_join()`
- B: `st_crop()`
- C: `st_intersection()`
- D: `left_join()`

Summary

- `st_crs()` : coordinates reference system
- `st_bbox()` : spatial extent
- `st_join()` : spatial left join
- `st_intersection()` : intersection join
- `st_crop()` : crop
- `st_union()` : union
- `st_difference()` : difference
- `st_area()` : area
- `st_distance()` : distance
- `st_length()` : length
- `st_centroid()` : find centroid
- `st_buffer()` : buffer area

Your turn!

Exercises in: *"exo_explore_saint_louis.Rmd"*

- Start from title `# **Exercises for Manipulating vector data: Spatial manipulations**` until the end !

Thanks for joining this tutorial!

Sébastien Rochette, Dorris Scott, Jakub Nowosad

@statnmap, @Dorris_Scott, @jakub_nowosad

Material of this course is on Github (with answers):
statnmap/user2020_rspatial_tutorial