🌍🌎🌏 Rmapping 🌍🌎🌏

PhD Toolbox
R introductory course - stream II

Martino ADAMO && Matteo CHIALVA

What does “mapping” means

Georeferencing of biological data (or mapping for the friends) is part of **topography**

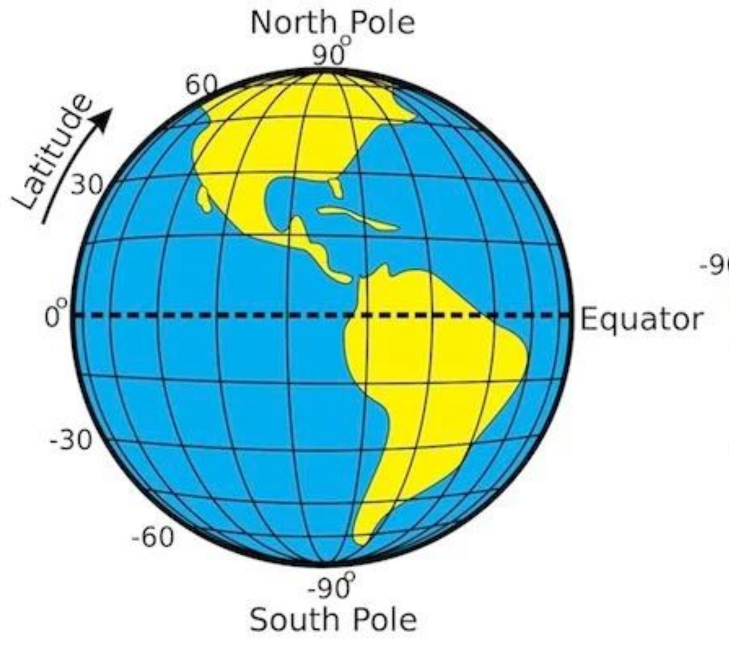
Definition 1:

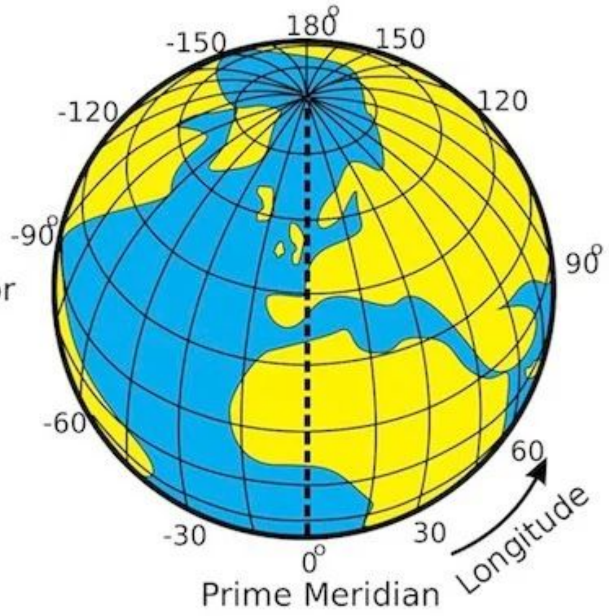
Topography is the study of the forms and **features** of land surfaces. The topography of an area may refer to the land forms and features themselves, or a description or depiction in maps.

Definition 2:

Topography is a field of geoscience and planetary science and is concerned with local detail in general, **including** not only relief, but also **natural**, artificial, and cultural **features** such as roads, land boundaries, and buildings.

Georeferencing





Each site on our sub-spheric planet can be individuated using an X and an Y value called coordinates.

X = longitude

Y = latitude

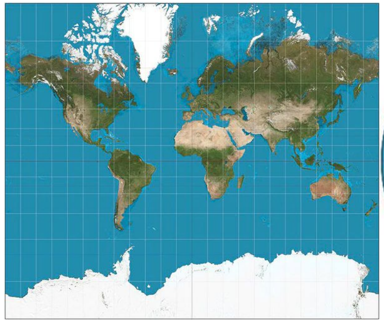
Coordinate systems are many and are based on several different models, each of them has its pros and its cons

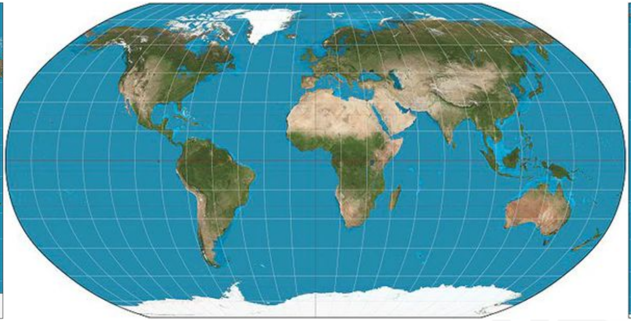
Basically it is difficult to PROJECT on a plane something spheric without distortions

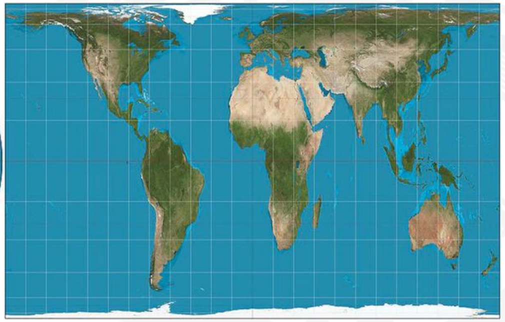
Georeferencing

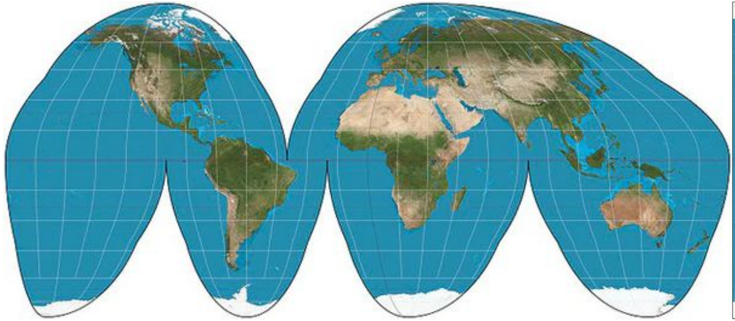
Projection are all wrong!

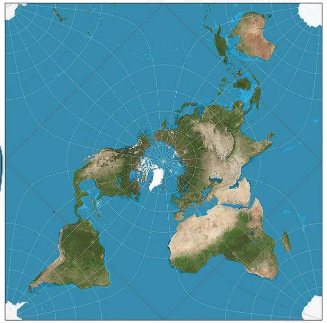
choose the best for your aim

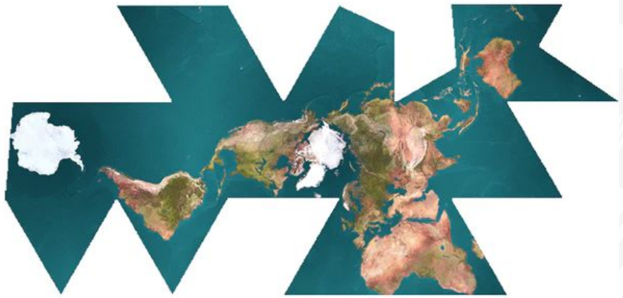

Mercator


Robinson


Gall-Peters


Goode’s Homolosine


Quincuncial

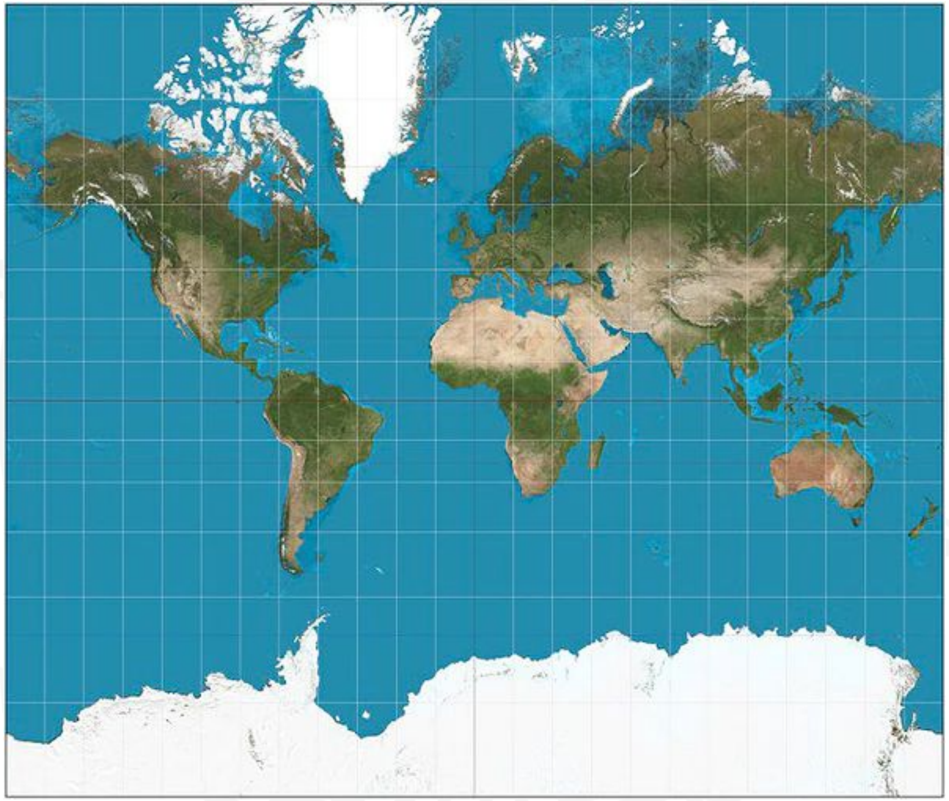

Dymaxion Map

WGS84

The World Geodetic System 1984 (or EPSG:4326) is a global geographic coordinates system based on a reference ellipsoid (or geoid).

Geoids are mathematically defined surface that approximates the Earth shape. Using ellipsoid distortions are partially corrected.

Usually maps based on the **WGS84** system are plotted using the **UTM (Mercator's) projection.**



WGS84

The Global Position Systems (GPS) and its rivals (GLONASS, Galileo and BeiDou) are all based on WGS84

This is a very good argument to use WGS84

!! careful !!

Italian national and regional data are often in other reference systems:

UTM ED50 33N (EPSG:23033)

Gauss-Boaga Roma 40 (ESRI:102094)

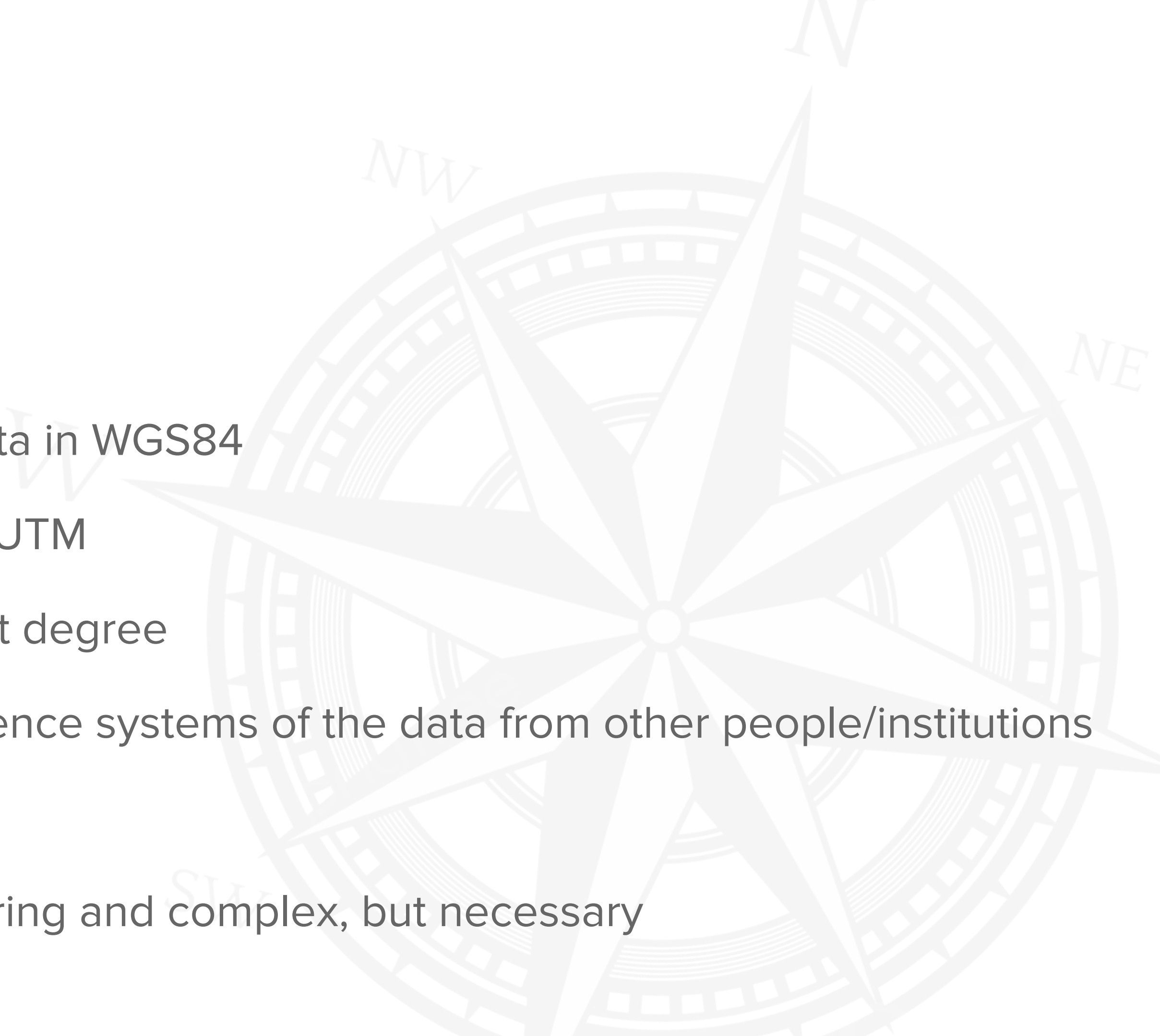
WGS84

lon	lat	coordinate system
7.549530	44.388654	WGS84
1384466.033970	4916154.933469	Gauss-Boaga Roma 40
-93387.969562	4942302.394422	UTM ED50 33N

moreover coordinates can be shown as:

- decimals
- degree minutes seconds
- degree minutes

🤯



WGS84

suggestions

- transform all your data in WGS84
- visualize all maps in UTM
- use decimals and not degree
- always ask the reference systems of the data from other people/institutions

data transformation is boring and complex, but necessary

R vs QGIS

R

- Output is valuable, but!
- Code recreates output
- Code is reusable instructions
- All analysis are possible

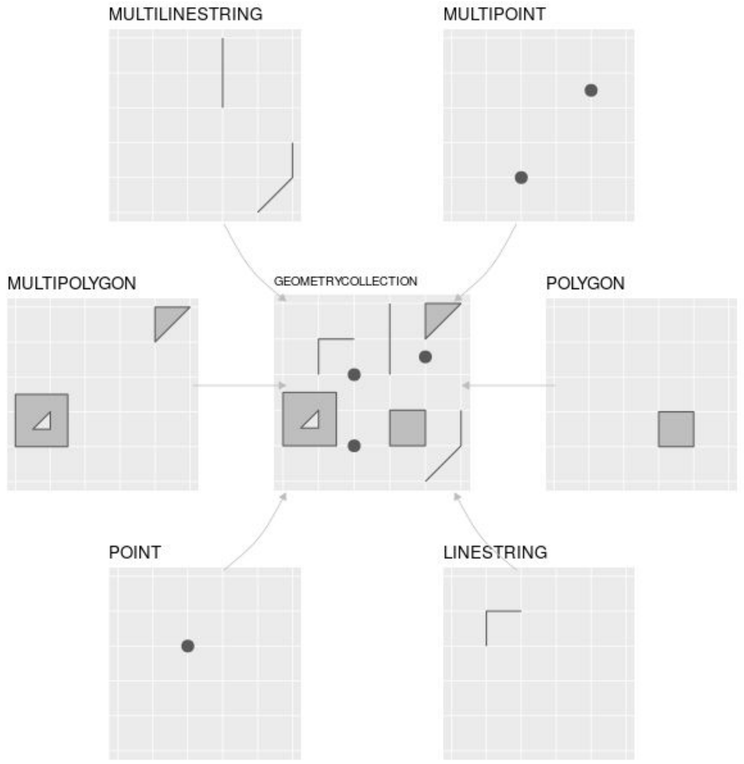
Best to analyze data

QGIS

- Output is valuable
- Workspace preserves settings
- Hard to explain how
- Several analysis are not as you want

Best to interactively visualize

What you can plot on a map



Examples:

points = a species observations

polygon = an habitat extension

linestring = a transect

raster vs vectors

What is a raster file?

Raster files are **images built from pixels** — tiny color squares that, in great quantity, can form highly detailed images such as photographs. The more pixels an image has, the higher quality it will be, and vice versa. The number of pixels in an image depends on the file type (for example, JPEG, GIF, or PNG).

What is a vector file?

Vector files use **mathematical equations**, lines, and curves with fixed points on a grid to produce an image. There are no pixels in a vector file. A vector file's mathematical formulas capture shape, border, and fill color to build an image. Because the mathematical formula recalibrates to any size, you can scale a vector image up or down without impacting its quality.

Resolution.

One of the main differences between raster and vector files is their resolution. The resolution of a raster file is referred to in **DPI** (dots per inch). If you zoom in or expand the size of a raster image, you start to see the individual pixels.

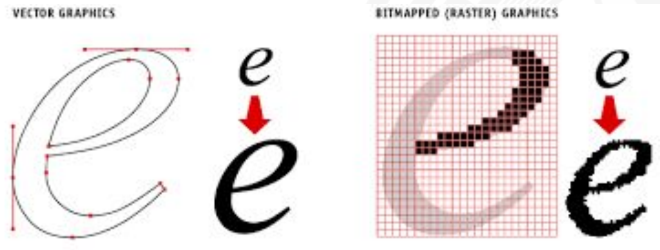
File sizes.

Raster files are generally larger than vector files. They can contain millions of pixels and incredibly high levels of detail. Their large size can impact device storage space and slow down page loading speeds on the web. However, you can compress raster files for storage and web optimization to make sharing faster and easier.

raster vs vectors

mathematical equations

lines, and curves with fixed points on a grid to produce an image



images built from pixels

higher resolution to smaller pixels

exactly the same concepts are applied to maps

vectorial maps == shapefiles (.shp)

raster maps == geotiff images (*.tiff)

Maps on R

Several different ways: I'll show you two options ... but they are plenty

Basic maps (with ggplot2)

- import map from CRAN
- plot countries and regions
- fill them by variables/factors
- plot points on a country map

Hardcore maps

- deal with rasters and shapefiles
- work with "terra"
  - crop
  - mask
  - extract
- plot them with or without ggplot2

mapping countries on ggplot2

We are going to use several specific packages:

- terra
- geodata
- tidyterra

Retrieving maps

Species distribution

GBIF

Spatial data

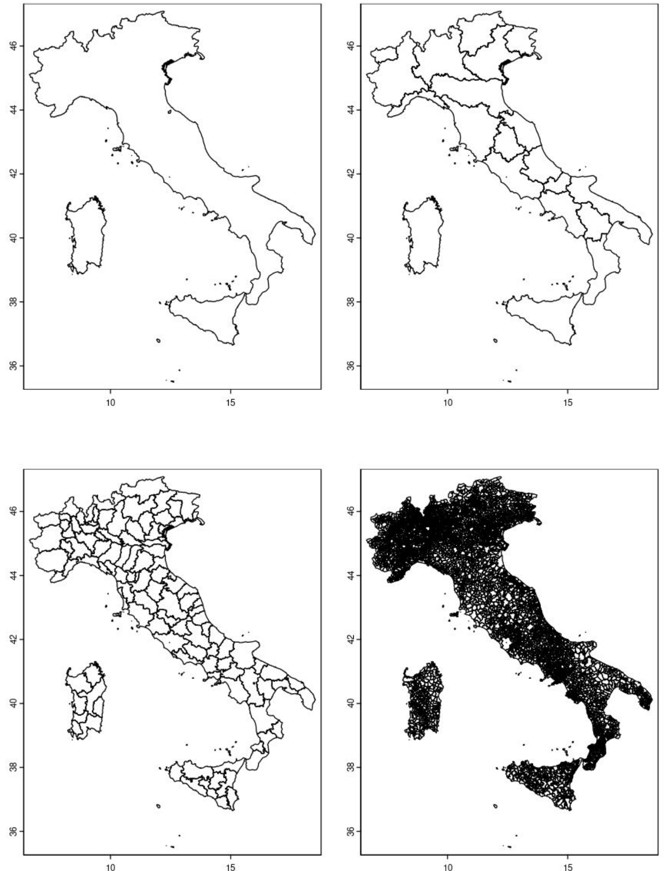
```
library(geodata)
```

- CMIP6 projected future climate data
- Elevation 30s (~1km) and 3s (~90m)
- GADM
- Global Landcover (ESA WorldCover 10 m 2020 - Copernicus)
- WorldClim 30s (~1km)
- SoilGrid (~250m)
- … and many others

mapping countries on ggplot2

The gadm() function:

```
library(tidyverse)
library(geodata)
library(terra)

IT0 <- gadm(country = "ITA", level=0, path = tempdir())
IT1 <- gadm(country = "ITA", level=1, path = tempdir())
IT2 <- gadm(country = "ITA", level=2, path = tempdir())
IT3 <- gadm(country = "ITA", level=3, path = tempdir())

par(mfrow = c(2,2))
plot(IT0) # country
plot(IT1) # region
plot(IT2) # province
plot(IT3) # city
```



mapping countries on ggplot2

inspecting spatial objects

```
print(IT2)
```

```
 class       : SpatVector
 geometry    : polygons
 dimensions  : 110, 13  (geometries, attributes)
 extent      : 6.630879, 18.52069, 35.49292, 47.09265  (xmin, xmax, ymin, ymax)
 coord. ref. : lon/lat WGS 84 (EPSG:4326)
```

OBJECT CLASS
SHAPE CLASS
DIM()
MAP EXTENT (OR LIMITS)
COORDINATES REF. SYSTEM

WHAT YOU CAN FIND INSIDE:

```
 names       :     GID_2 GID_0 COUNTRY   GID_1  NAME_1 NL_NAME_1   NAME_2 VARNAME_2 NL_NAME_2    TYPE_2 (and 3 more)
 type        :     <chr> <chr>   <chr>   <chr>   <chr>     <chr>    <chr>     <chr>     <chr>     <chr>
 values      : ITA.1.1_1   ITA   Italy ITA.1_1 Abruzzo        NA   Chieti        NA        NA Provincia
               ITA.1.2_1   ITA   Italy ITA.1_1 Abruzzo        NA L'Aquila    Aquila        NA Provincia
               ITA.1.3_1   ITA   Italy ITA.1_1 Abruzzo        NA  Pescara        NA        NA Provincia
```

mapping countries on ggplot2

inspecting spatial objects

```
> unique(IT2[["NAME_1"]])
                    NAME_1
1                  Abruzzo
5                   Apulia
11              Basilicata
13                Calabria
18                Campania
23          Emilia-Romagna
32   Friuli-Venezia Giulia
36                   Lazio
41                 Liguria
45               Lombardia
57                  Marche
62                  Molise
64                Piemonte
72                Sardegna
80                  Sicily
89                 Toscana
99     Trentino-Alto Adige
101                 Umbria
103          Valle d'Aosta
104                 Veneto
```

We can subset our spatial object basing on the features table. The function is subset:

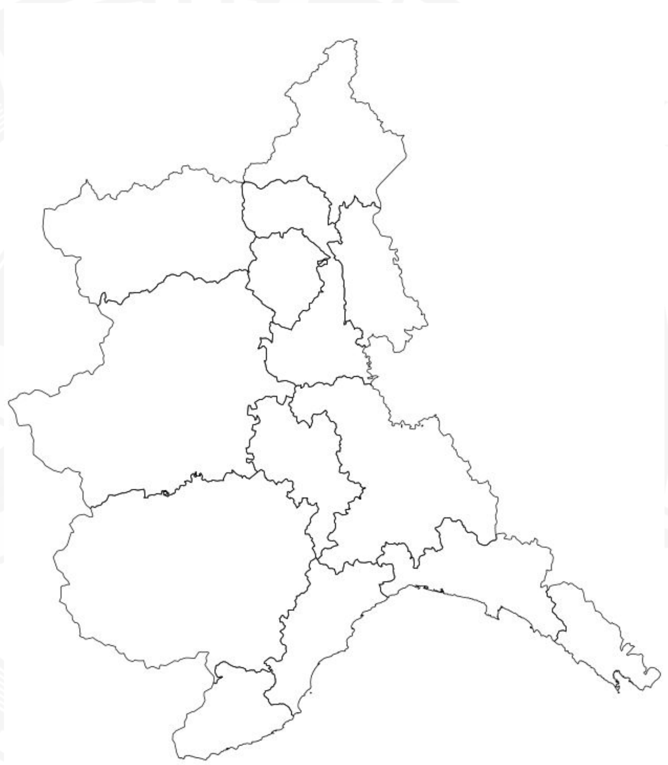
```
subset(SpatVect, Condition, Vector of columns to keep)
```

Let’s do it creating an object called NW including the three regions Piemonte, Liguria, Valle d’Aosta.

mapping countries on ggplot2

plot the map with ggplot! we need a new package called tidyterra

```
library(tidyterra)

ggplot() +
  geom_spatvector(data = NW, fill = NA, col = "black") +
  theme_void()
```



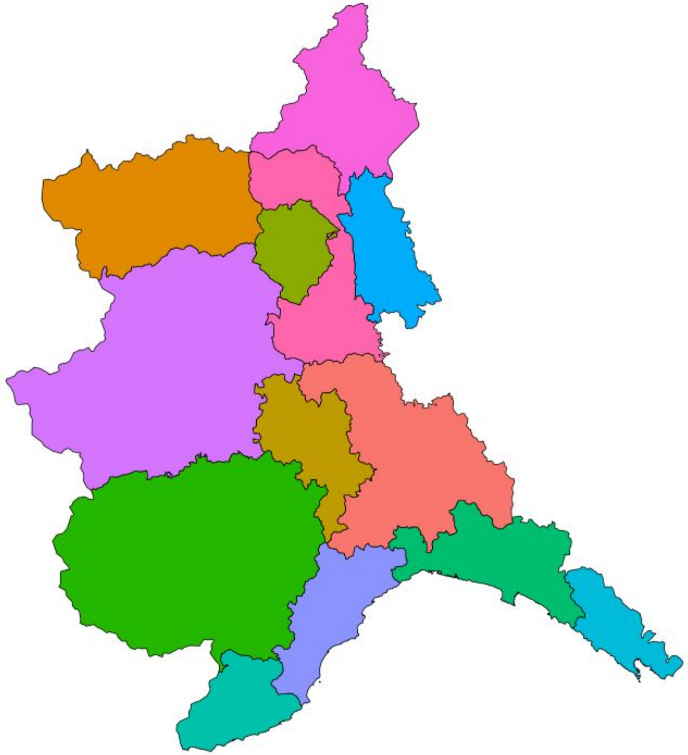
mapping countries on ggplot2

plot the map with ggplot! we need a new package called tidyterra

```
library(tidyterra)
```

we can fill the provinces by colour

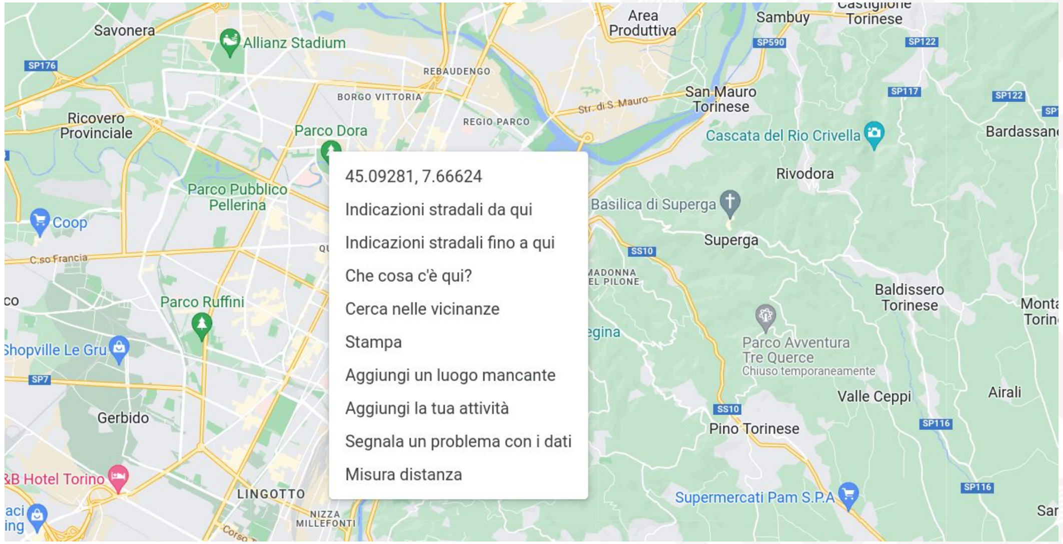
```
ggplot() +
  geom_spatvector(data = NW, aes(fill = NAME_2),
                  col = "black") +
  scale_color_brewer(palette = "Dark2") +
  theme_void() + theme(legend.position = "none")
```



mapping countries on ggplot2

```
cities <- data.frame(cities = c("Aosta","Novara","Torino","Alessandria","Cuneo","Genova"),

lat = c(45.73796224152826, 45.457835274700855,45.0659968909342,44.90219559193821,44.379342903855886,44.41116507155628),

lon = c(7.29944187639308, 8.620470913953753,7.694126932260661,8.602656606613502, 7.5293445893633315, 8.954489176583474))
```

A dataframe containing coordinates corresponding to cities … you can extract them directly from GoogleMaps …

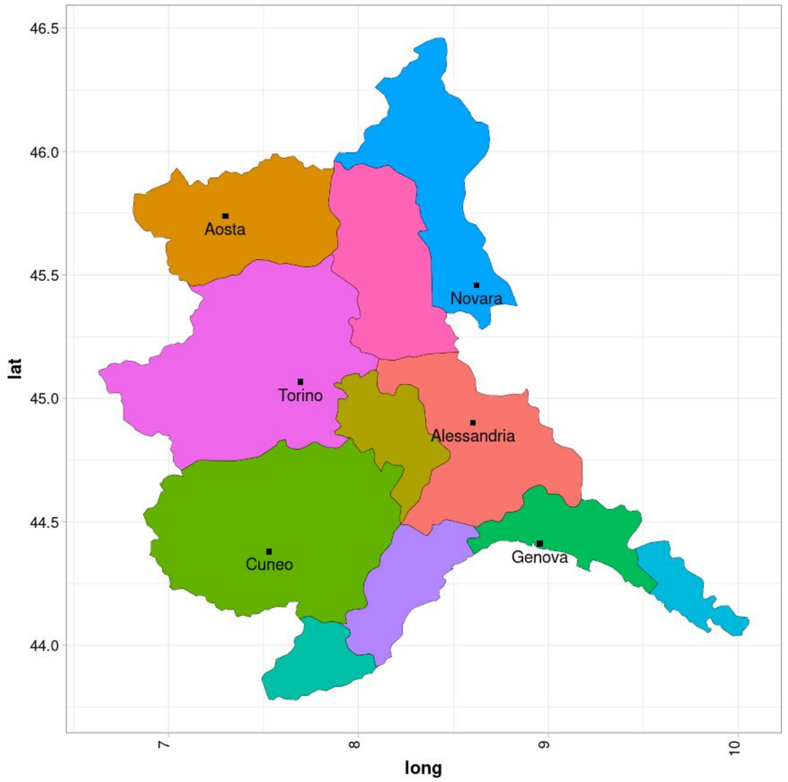


45.09281, 7.66624
Indicazioni stradali da qui
Indicazioni stradali fino a qui
Che cosa c'è qui?
Cerca nelle vicinanze
Stampa
Aggiungi un luogo mancante
Aggiungi la tua attività
Segnala un problema con i dati
Misura distanza

mapping countries on ggplot2

Exercise:

plot cities points and their names over the NW-Italy map



mapping countries on ggplot2

Exercise:

plot cities points over the NW-Italy map

```
ggplot() +
  geom_spatvector(data = NW, aes(fill = NAME_2), col = "black") +
  scale_color_brewer(palette = "Dark2") +
  geom_point(data = cities, aes(x = lon, y = lat), color = "black", shape = 15) +
  geom_text(data = cities, aes(x = lon, y = lat, label = cities), vjust = 1.5) +
  theme_void() + theme(legend.position = "none")
```

mapping countries on ggplot2

sometime it could be great to associate quantitative data to your map:

for example from the ISTAT website it is easy to obtain several statistics about agronomic production …

I created a specific dataframe called Frumento.tsv

Exercise:

generate a map where provinces are filled with the wheat production by year using a gradient. This exercise is going to summarize several different steps we explained in the previous two lessons!

you can use the function **merge()** to associate the Frumento dataframe with the NW SpatVector

https://github.com/mchialva/PhDToolbox2026/tree/main/Datasets

WORKING WITH RASTERS

```
library(geodata)

library(terra)

library(tidyterra)
```

these are the three packages we are going to use in the next complex few steps:

WORKING WITH RASTERS

using geodata we download rasters from four ecologically meaningful abiotic factors

```
> h3 <- elevation_30s("Italy", path = tempdir(), mask=FALSE)

> h3

class       : SpatRaster

dimensions  : 1428, 1464, 1  (nrow, ncol, nlyr)-> pixels by row and by column

resolution  : 0.008333333, 0.008333333  (x, y)-> corresponding to 30s!

extent      : 6.5, 18.7, 35.3, 47.2  (xmin, xmax, ymin, ymax)

coord. ref. : lon/lat WGS 84 (EPSG:4326)

source      : ITA_elv.tif

name        : ITA_elv

min value   :     -73

max value   :    4498
```

WORKING WITH RASTERS

using geodata we download rasters from four ecologically meaningful abiotic factors

```
> tavg <- worldclim_country("Italy", "tavg", path= tempdir(), res = 0.5)

> tavg

class       : SpatRaster

dimensions  : 1500, 1500, 12  (nrow, ncol, nlyr)-> 12 layers! Monthly data

resolution  : 0.008333333, 0.008333333  (x, y)

extent      : 6.5, 19, 35, 47.5  (xmin, xmax, ymin, ymax)-> extent is different!
…
```

these are monthly data!

WORKING WITH RASTERS

To simplify the analysis is better to create a single ANNUAL layer.

`tavg[[1]]` = **January**, `tavg[[2]]` = **February**, `tavg[[1]]` = **March**, …

```
tavg_mean <- mean(tavg)

names(tavg_mean)                     # always check the variable name!

names(tavg_mean) <- "annual_tavg"    # always check the variable name!

tavg_mean
```

exercise: download and convert also precipitation and wind speed!

WORKING WITH RASTERS

we need that all rasters share the same resolution, coordinate system and extent

```
> stack <- c(tavg_mean, prec_mean, wind_mean, h3) # does not work!
Error: [rast] extents do not match


>stack1 <- c(tavg_mean, prec_mean, wind_mean)# create a stack with only the WorldClim guys
>stack1_crop <- crop(stack1, ext(h3))# Crop them on the base of the h3 extent
```

ext() is the function to extract the extent value of a map

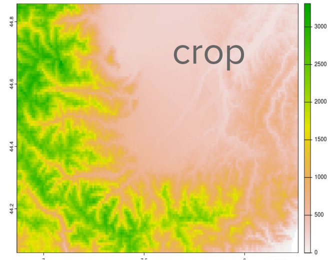
crop() is the function to cut the rectangle defined in the raster extent

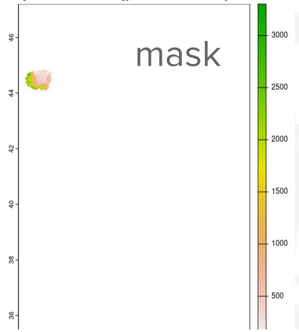
```
>stack <- c(stack1_crop, h3) # Then you can stack all!
```

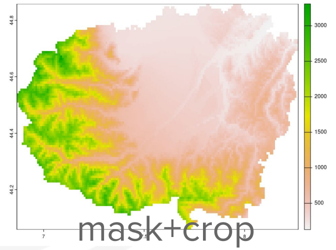
WORKING WITH RASTERS

The two functions **crop()** and **mask()** are similar but different!

```
h3_crop <- crop(h3, ext(subset(NW, NW$NAME_2 == "Cuneo")))

plot(h3_crop)


h3_mask <- mask(h3, subset(NW, NW$NAME_2 == "Cuneo"))

plot(h3_mask)


h3_mask_crop <- crop(mask(h3, subset(NW, NW$NAME_2 == "Cuneo")),

      ext(subset(NW, NW$NAME_2 == "Cuneo")))

plot(h3_mask_crop)
```







WORKING WITH RASTERS

using the Italy shape obtained by GADM we can mask our raster to see only the Italy shape

```
> IT <- gadm("Italy", level=0, path= tempdir(), version="latest", resolution=1)
```

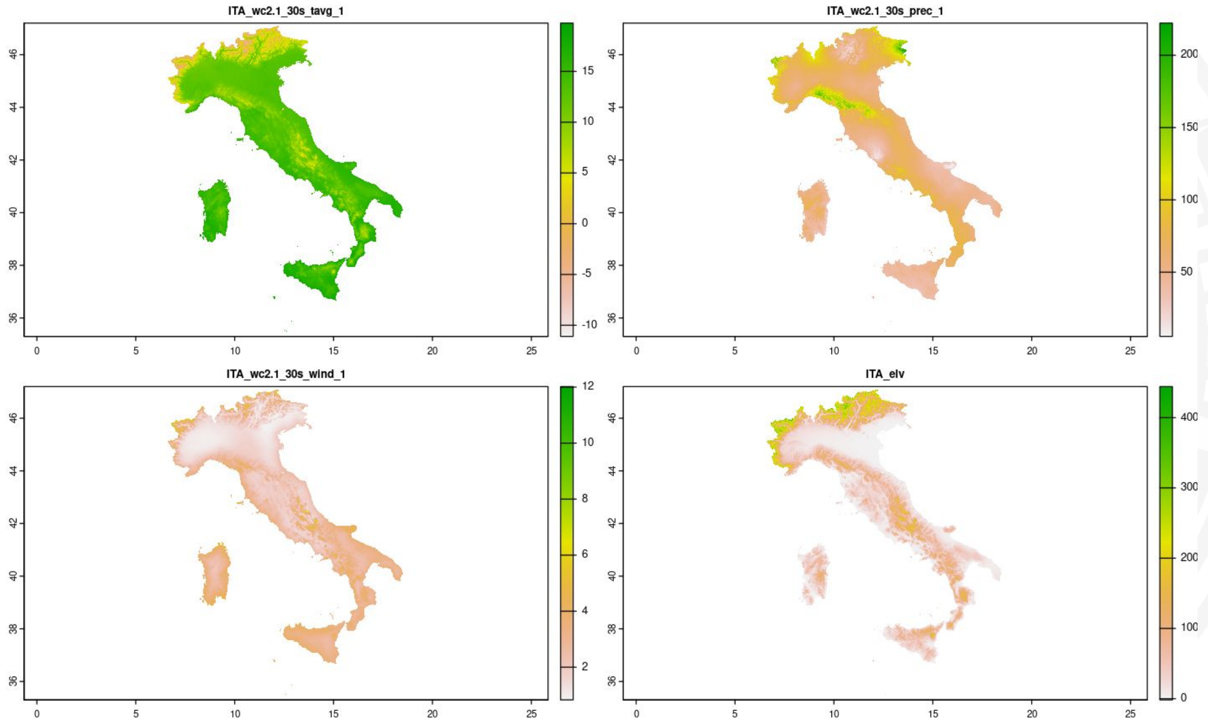
level 0 = country border

level 1 = regions

```
...

> stack_mask <- mask(stack, IT) # mask the stack using the Italy shape

plot(stack_mask)

stack_mask
```

WORKING WITH RASTERS

```
plot(stack_mask)
```



WORKING WITH RASTERS

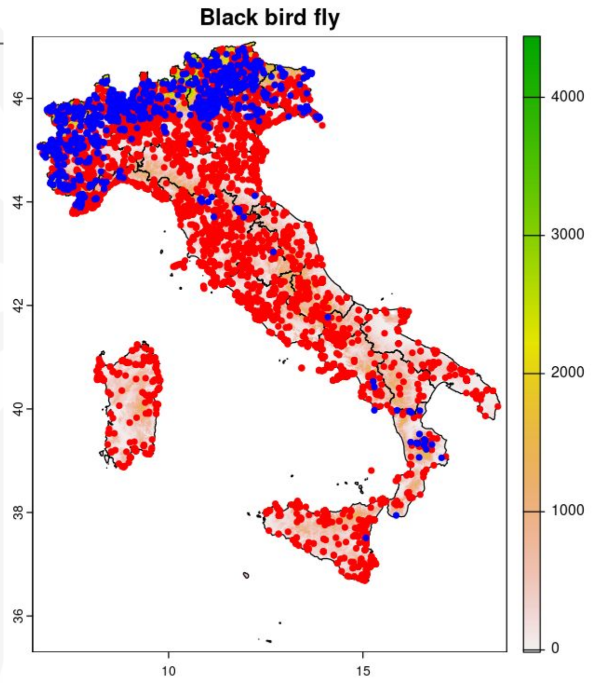
```
> stack_mask

class       : SpatRaster

dimensions  : 1428, 1464, 4  (nrow, ncol, nlyr)

resolution  : 0.008333333, 0.008333333  (x, y)

extent      : 6.5, 18.7, 35.3, 47.2  (xmin, xmax, ymin, ymax)

coord. ref. : lon/lat WGS 84 (EPSG:4326)

source(s)   : memory

names       : ITA_wc2~_tavg_1, ITA_wc2~_prec_1, ITA_wc2~_wind_1, ITA_elv

min values  :       -11.10000,          5.2500,       0.8333333,     -73

max values  :        19.84167,        230.9167,      12.0166667,    4442
```

PLOT RASTERS

Load the birds.tsv containing occurrences of two species (Great tit and Woodpecker)

This is the Rbase method to plot maps … I like it because it is raw and fast

```
# plot dots over the maps METHOD1 -------------------------------------------------

plot(stack2[["ITA_elv"]], main = "Black bird fly")

plot(IT1, add = TRUE)

points(birds[which(birds$acceptedScientificName == "Parus major"),2:3],

col = "red", pch = 20)

points(birds[which(birds$acceptedScientificName == "Dryocopus martius"),2:3],

col ="blue",pch = 20)
```



PLOT RASTERS

We can extract raster's data corresponding to our occurrences!

STEP1

verify the occurrences!

```
birds2 <- subset(birds, coordinateUncertaintyInMeters <= 1000)
```

STEP2

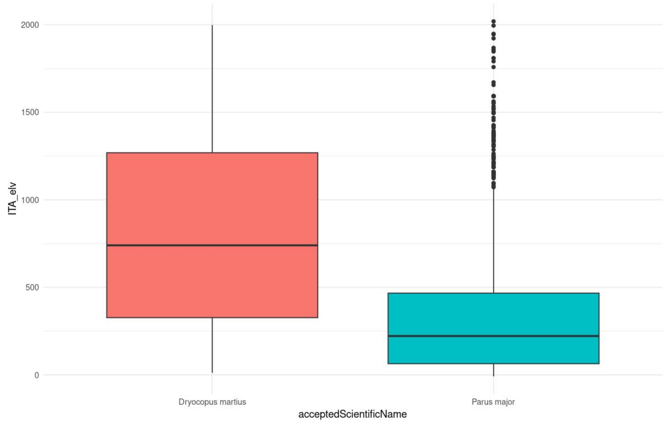
extraction using the terra package

```
feat <- extract(stack2, birds2[,2:3]) # optionally xy = TRUE
pred <- cbind.data.frame(birds2, feat) # association with original data
pred <- na.omit(pred) # deleting NAs
```

PLOT RASTERS

Extracted data are useful to calculate whatever you want, models, stats, … and so on

```
ggplot(pred, aes(x=acceptedScientificName, y = ITA_elv)) +
    geom_boxplot(aes(fill = acceptedScientificName)) +
    theme_minimal() + theme(legend.position="none")
```



Black woodpecker usually lives at higher elevations than *Parus major* … blah blah blah

PLOT RASTERS

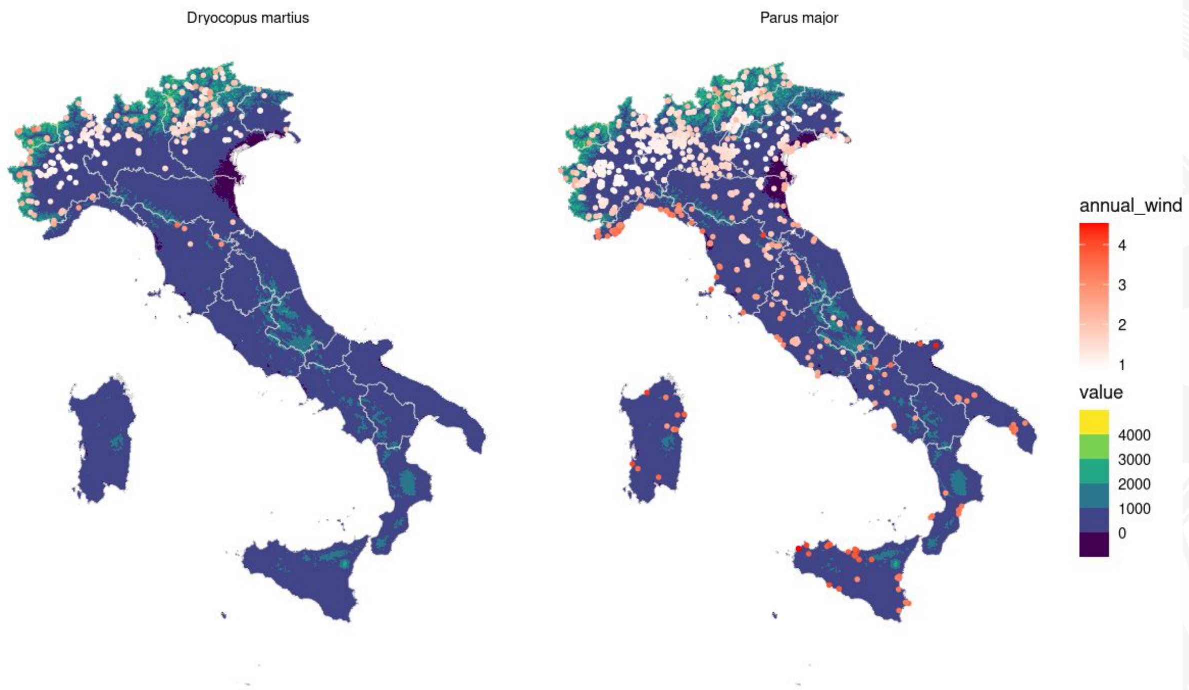
FINAL EXERCISE

using the **geom_spatraster()** function of **tidyterra** plot over the elevation map of Italy:

- regional boundaries
- species occurrences colored by wind speed

PLOT RASTERS

FINAL EXERCISE


Dryocopus martius
Parus major
annual_wind
4
3
2
1
value
4000
3000
2000
1000
0