Available on

GIS Programming Project

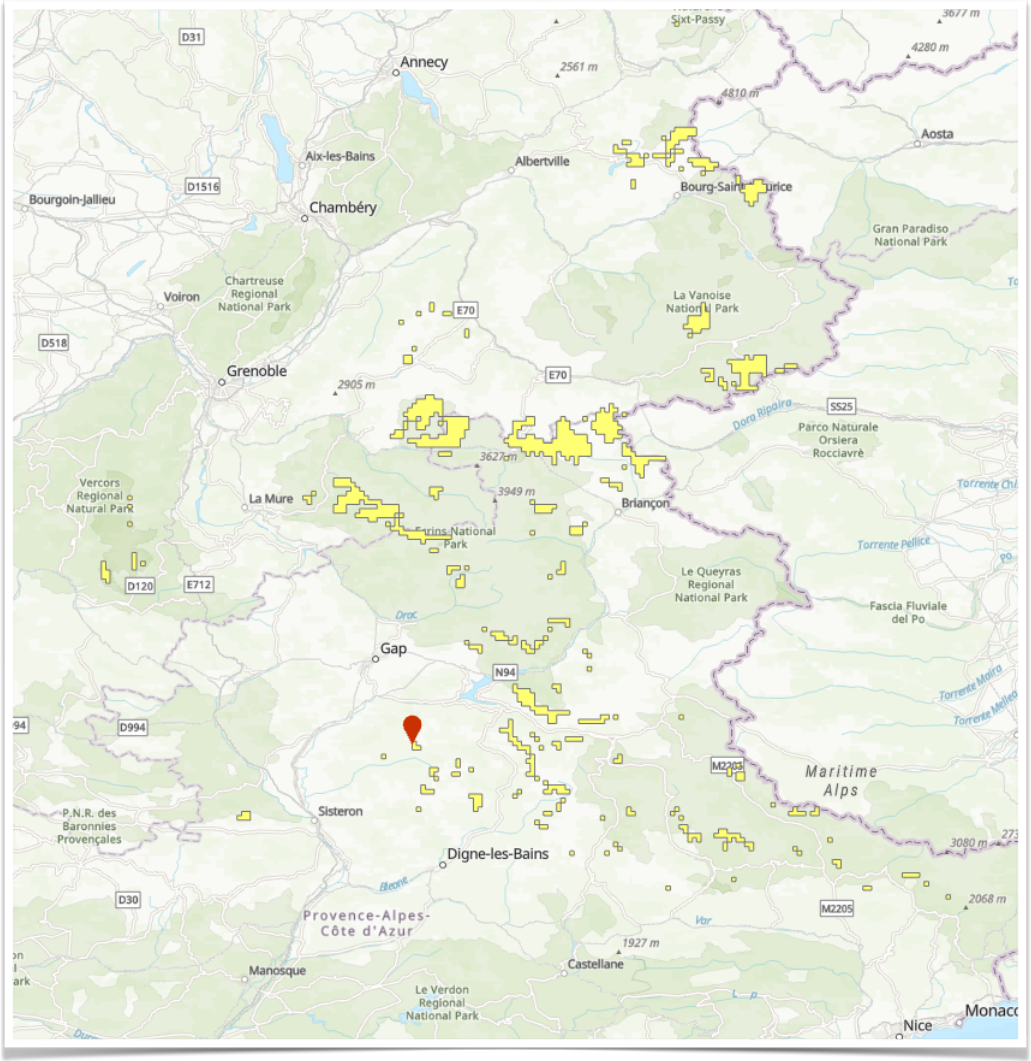
Find a location for a new astronomical observatory in France

Author: Maelle Sannier
Instructor: Naci Dilekli
Spring 2020

Objective of the project

This project is aimed to find a suitable place to implant **a new astronomical observatory in France**. It takes as inputs **the altitude** in France**, the luminous pollution** related to big cities, **existing** French **observatories** and the **mean value of cloudy days** per year.

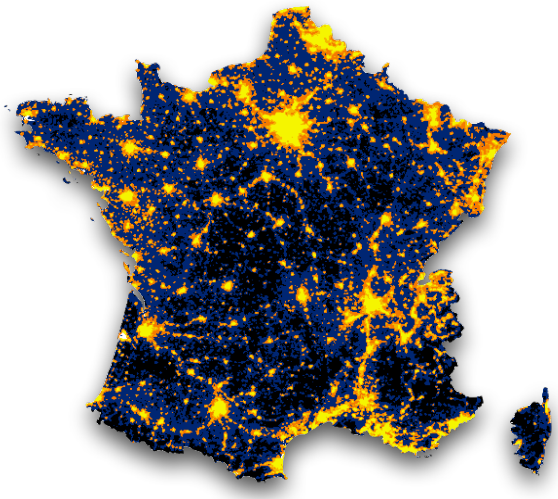
By comparing all these layers, the program would be able to give as an output some places likely to be chosen to implant a new observatory.

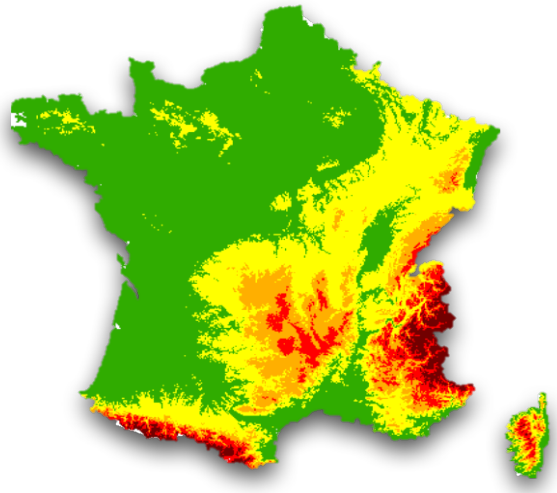


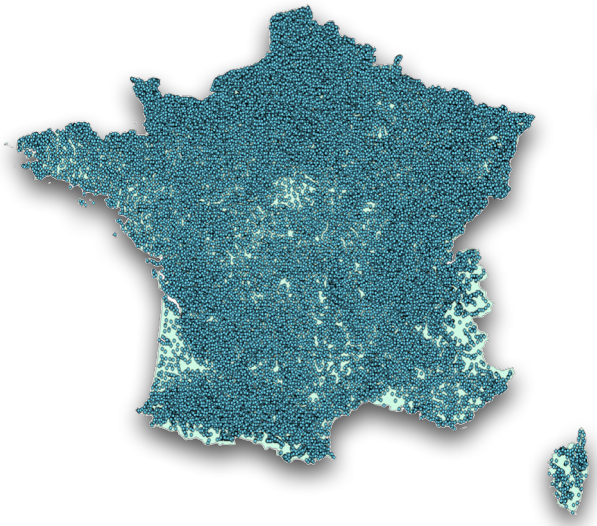
Data sources

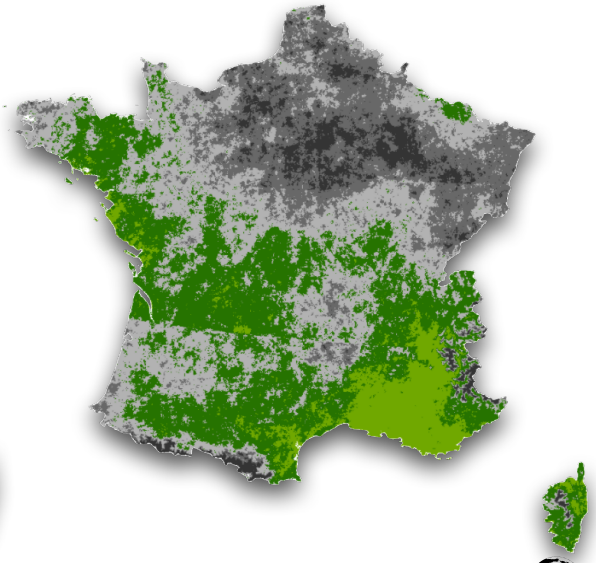
- Luminous Pollution
- Altitude
- Cloudiness
- French cities location and population
- Existing observatories

OBJECTID	NOM	Latitude	Longitude
1	Observatoire de Nice	49,7275	7,29916667
2	Observatoire des Pises	44,0394444	3,50361111
3	Observatoire du Mont Ventoux	44,1666667	5,28333333
4	Tour de la Babotte	43,6063889	3,8775
5	Observatoire de Toulouse	43,6122222	1,46277778
6	Observatoire de Paris	48,8363889	2,33638889
7	Observatoire de Paris/Meudon	48,805	2,23111111
8	Station de radioastronomie de Nança	47,3805556	2,195
9	Observatoire de la Côte d'Azur	43,7233333	7,30166667
10	Cerga	43,7530556	6,9225







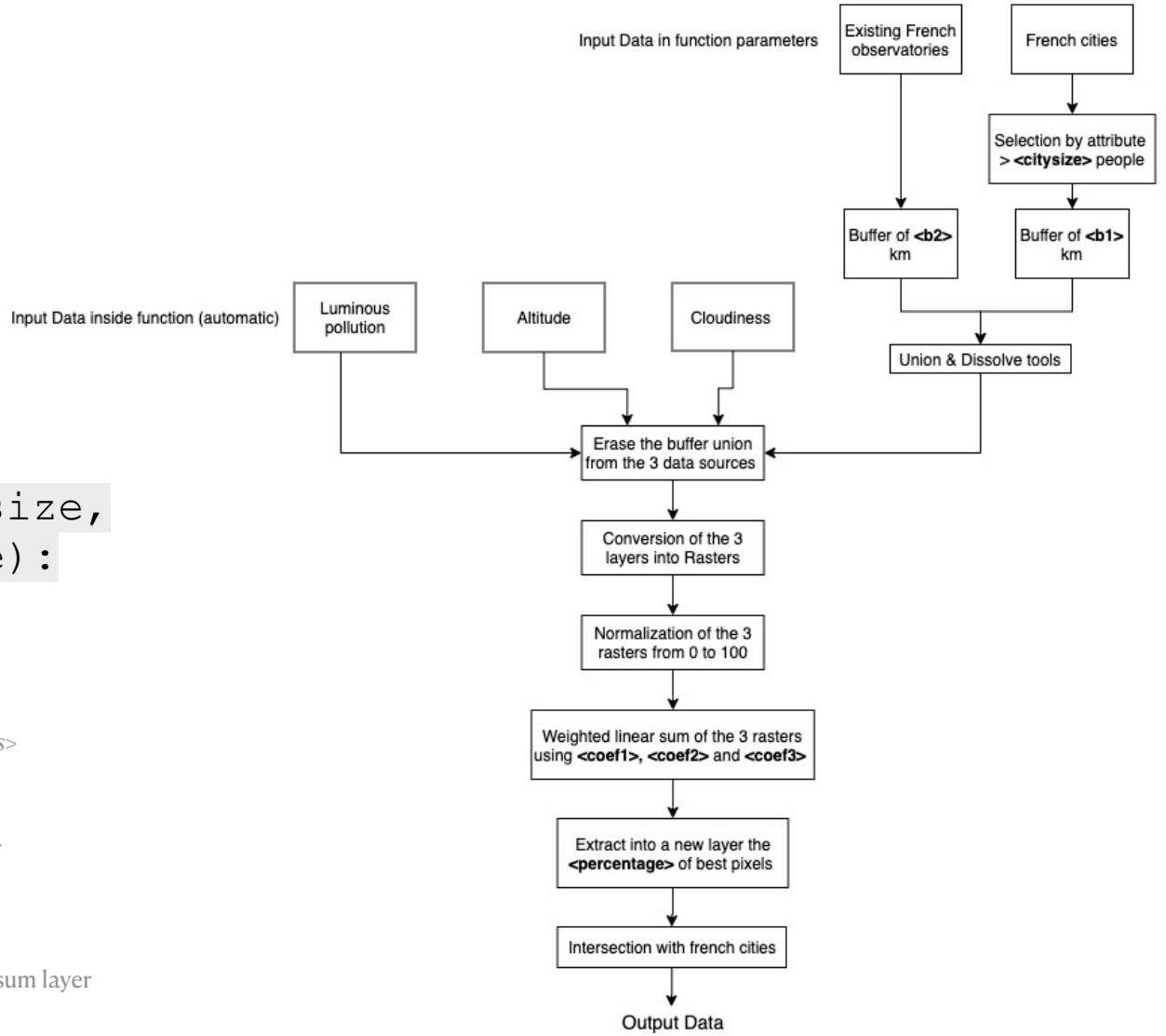


Methods

Flowchart presentation

```
def Find_Observatories
(workspace,fc1,b1,fc2,b2,citysize,
  coef1,coef2,coef3,percentage):
```

workspace = storage space defined by the user *e.g <C:/Observatories>*
fc1 = first layer to use to create buffers *<frenchville.shp>*
b1 = Radius Buffer in Kilometers for 1rst feature class *<30 Kilometers>*
fc2 = second layer to use to create buffers *<Observatories.shp>*
b2 = Radius buffer in kilometers for 2nd feature class
citysize = We want to avoid cities over <citysize> inhabitants *e.g <50000>*
coef1 = coefficient in the weighted sum for Crit1 *e.g <3>*
coef2 = coefficient in the weighted sum for Crit2
coef3 = coefficient in the weighted sum for Crit3
percentage = percentage between 0 and 100 of best pixels of the weigthed sum layer that the user wants to keep *e.g <10>*



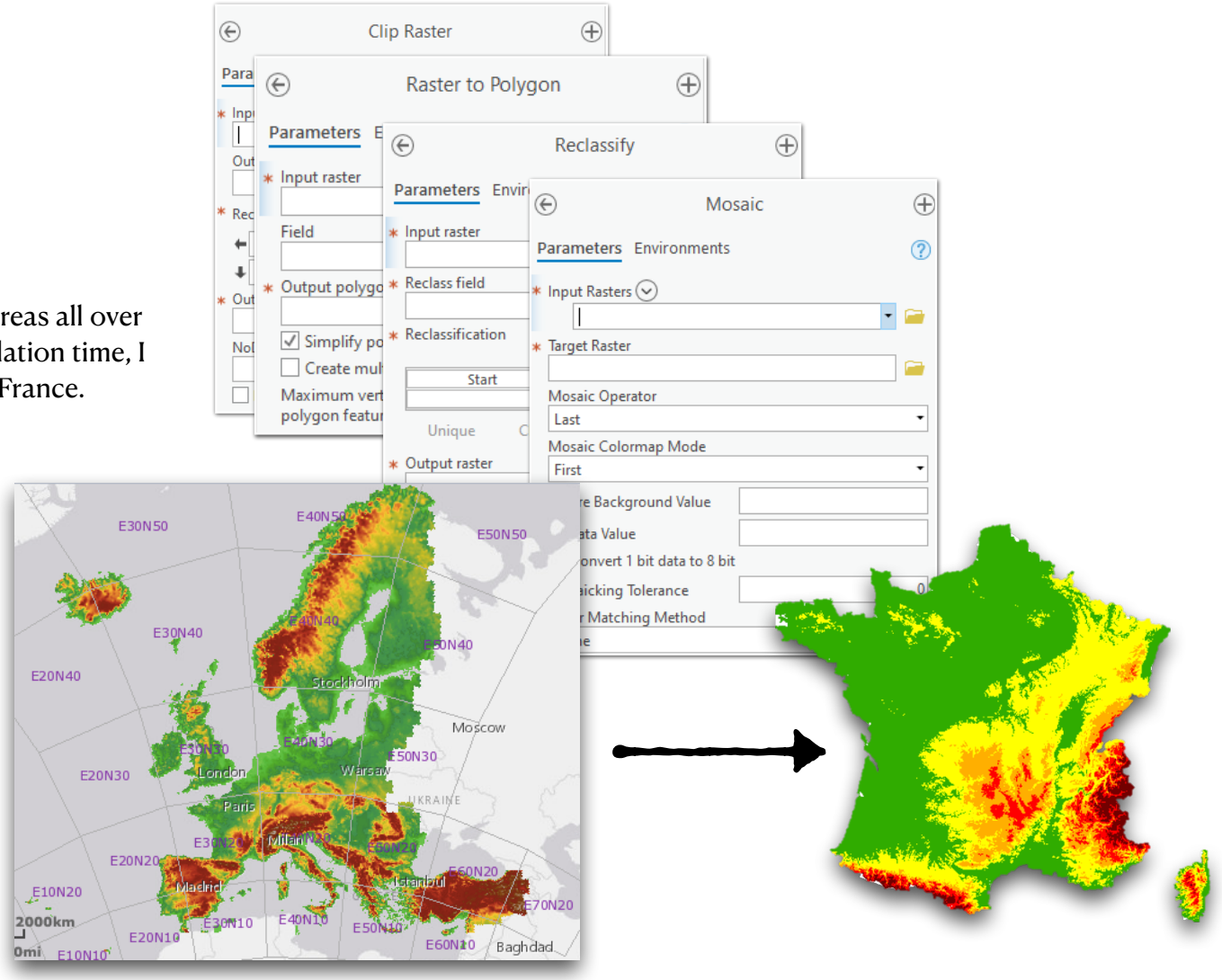
Methods

Previous work

The initial data sources was extended to very large areas all over Europe and all over the world. Thus, to reduce calculation time, I had to crop those data to the studied territory, France.

For that, I used the following tools :

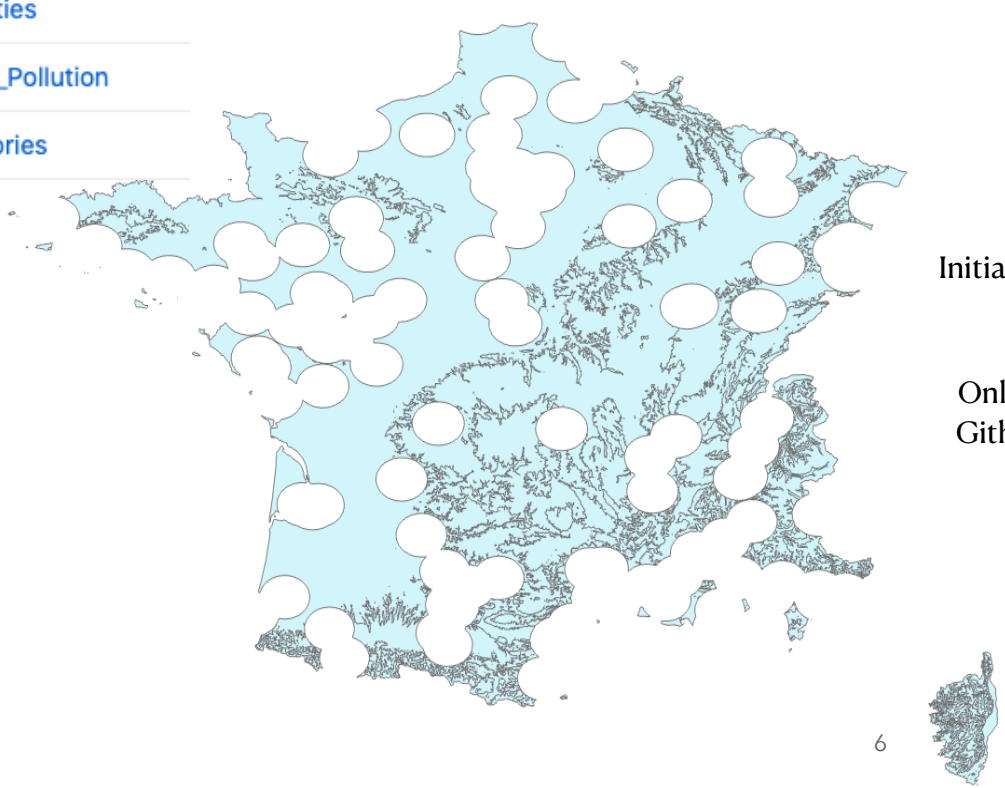
- Clip Raster
- Raster to Polygon
- Mosaïc
- Reclassify
- Display XY Data
- Add Field
- ...



Data available on github

maellesannier / EPF-RUDN-Observatory

- Altitude
- Cloudiness
- French_cities
- Luminous_Pollution
- Observatories



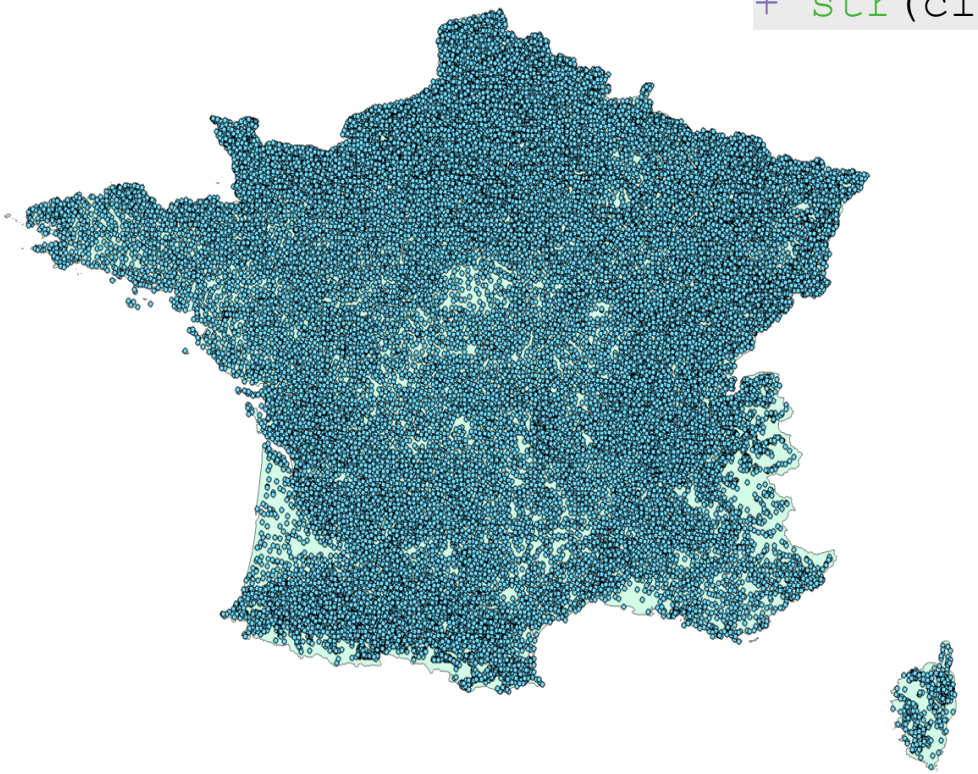
Initial clipped raster already vectorized to gain time calculation

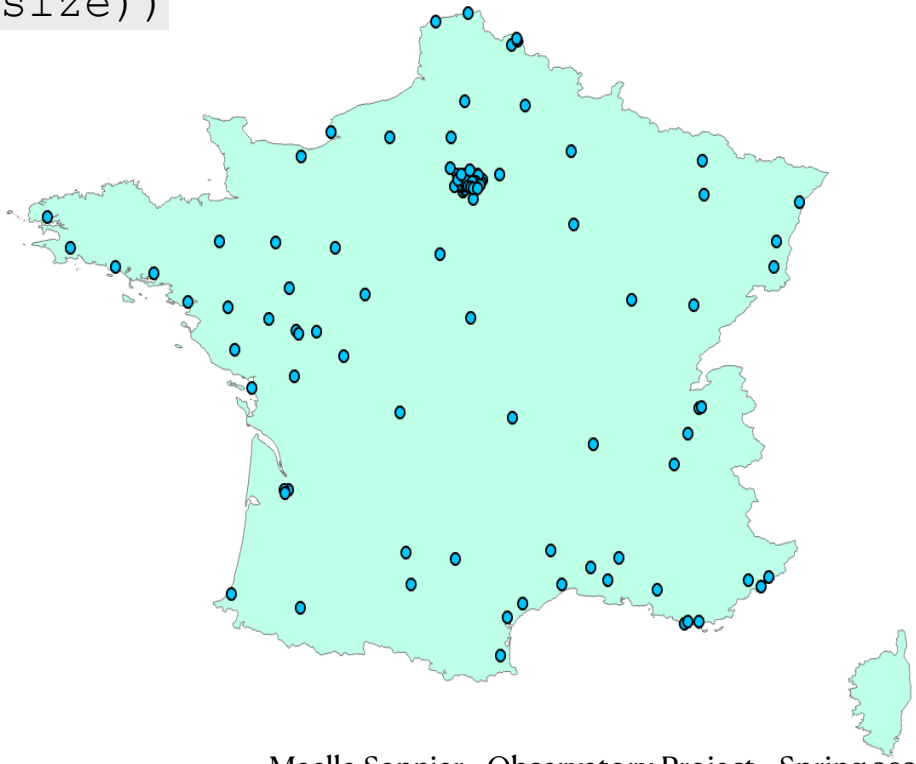
Only the Vectorized rasters are available in my repository on Github for you to gain calculation time. Indeed, converting a raster into a vector can take a long time.

Methods

Select by Attribute

```
arcpy.SelectLayerByAttribute_management(fc1,'NEW SELECTION', "PTOT >"
                + str(citysize))
```



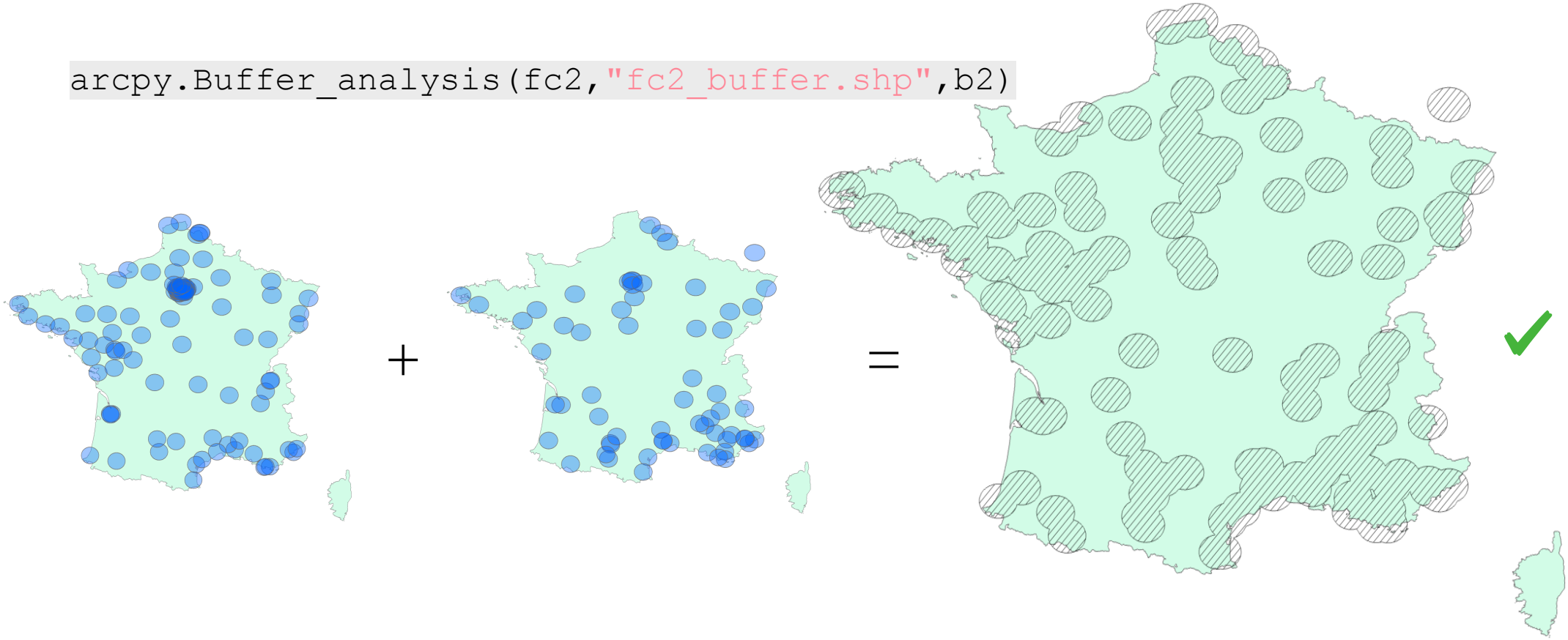


Methods

Buffers creation

buffers are the areas to avoid:

```
arcpy.Buffer_analysis(fc2,"fc2_buffer.shp",b2)
```

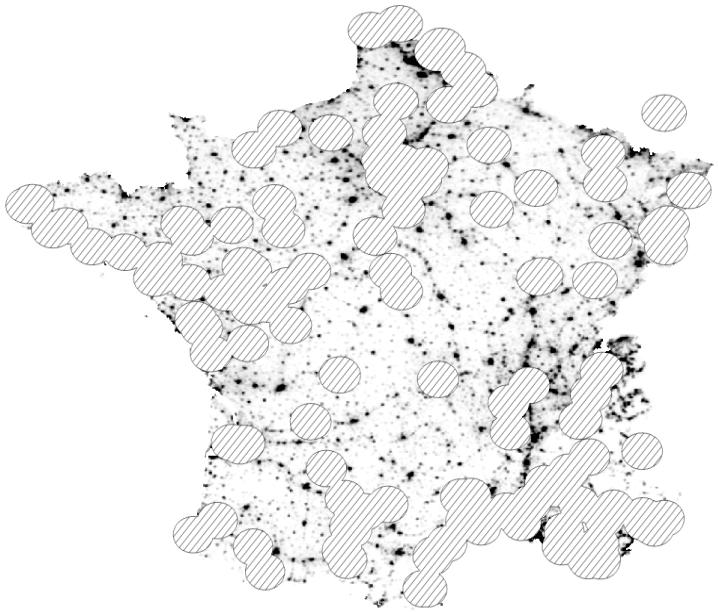


Methods

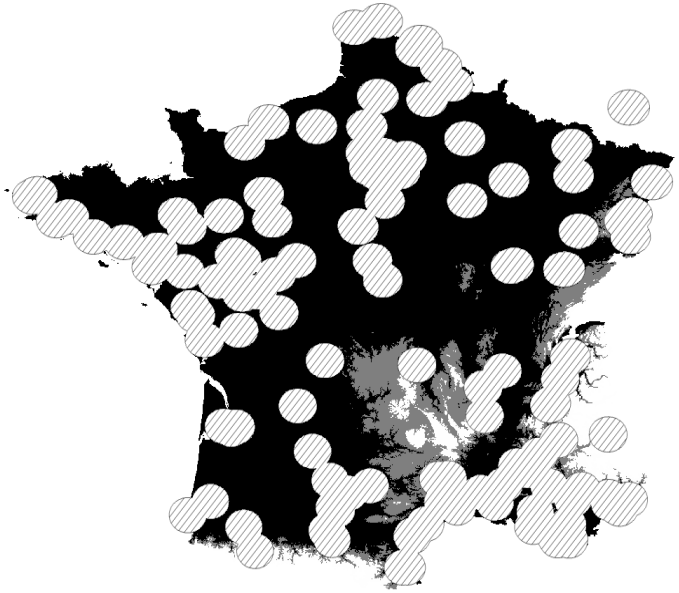
Erase tool

White represents the most wanted areas

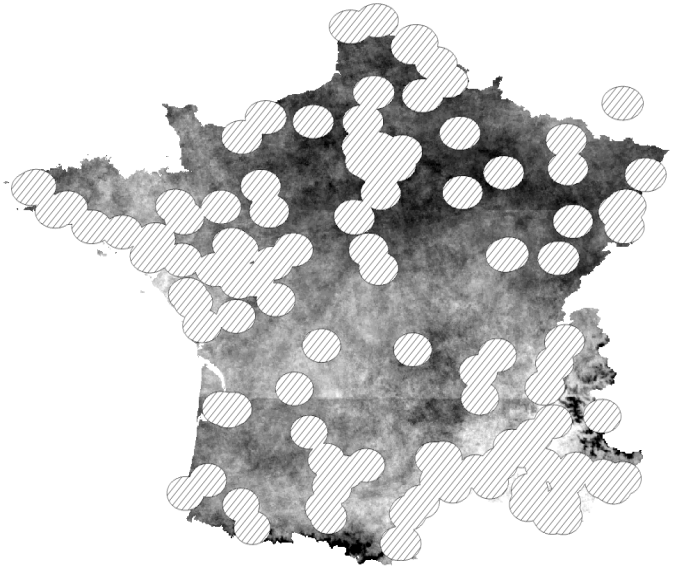
```
arcpy.Erase_analysis("Lum_Vector.shp","Buffers_Dissolved.shp","Crit1VctEra")
```



Luminous pollution



Altitude



Cloudiness

Methods

Normalization of rasters

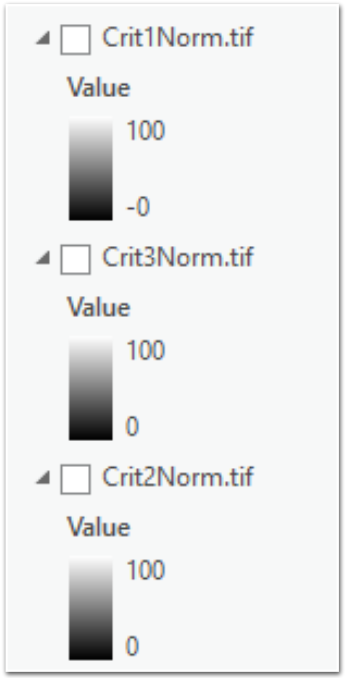
```
arcpy.Raster(workspace + "/Luminous_Pollution" + "/" + "Crit1Rst")

Normraster =(layer - layer.minimum) / (layer.maximum - layer.minimum) *100
```

range

Value between 0 and 1

Value between 0 and 100

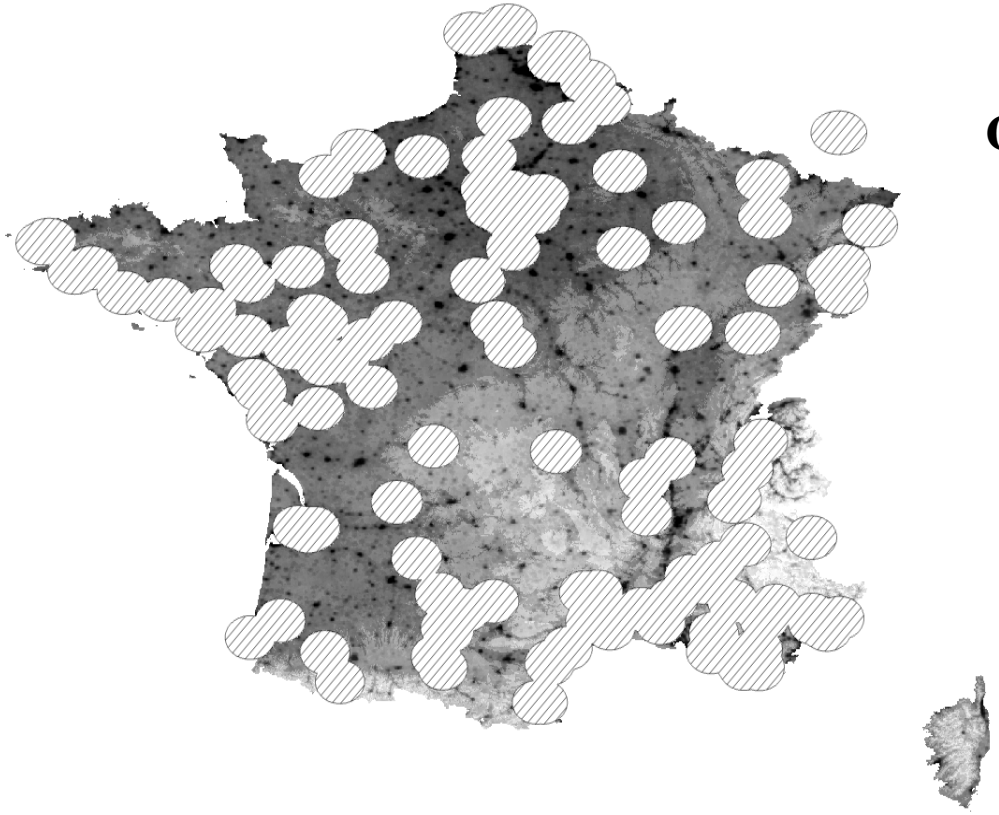


White represents the most wanted areas, « 100 » represents the higher values

Methods

Weighted sum

```
arithmeticlayer = coef1 * Crit1Norm + coef2 * Crit2Norm + coef3 * Crit3Norm
```



Goal:

Linear combination to add all the rasters with different coefficients in aim to make some criteria more important than others.

Thus, one can look at the pixels which get the highest values : these are the ones which best satisfy all the criteria.

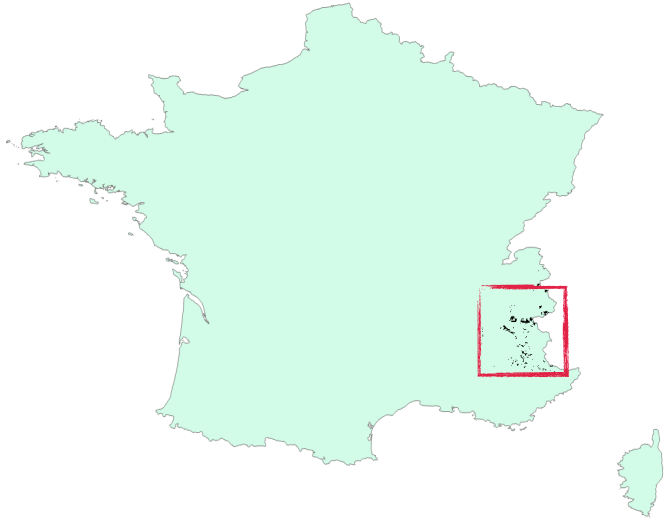
Methods

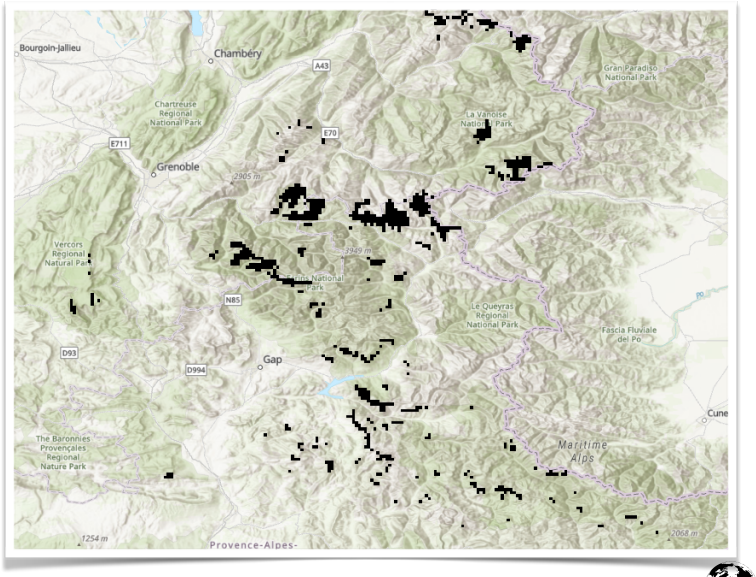
Keep only best pixels

```
condition = (layer.maximum - layer.minimum ) / (100/
percentage)

treshold = layer.maximum - condition

arcpy.sa.Con(arithmeticlayer,1,"","Value >" +
str(treshold))
```

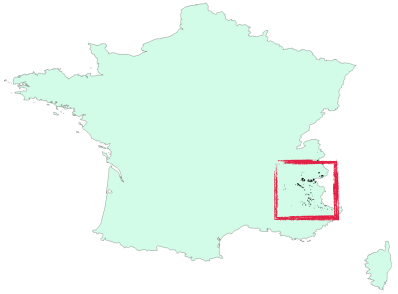




Goal: Vizualize the pixels that have accumulated the highest values

Methods

Intersection with little villages



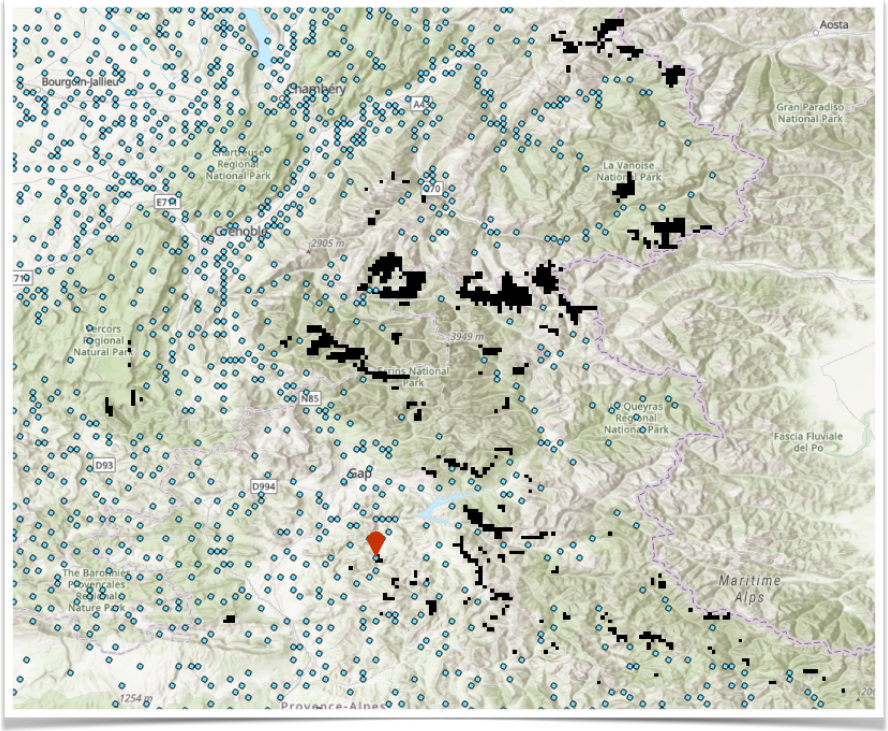
```
arcpy.Intersect_analysis(["BestpixelsVct.shp","frenchville.shp"],"FinalResult.shp")
```

Goal:

We intersect the best pixels with the cities layer so that the new observatory is implanted (near or) in a little village.

Indeed, we want to avoid a maximum the luminous pollution but the area for the observatory need to be accessible by the workers on a daily basis. Preference is given to little villages. By chance, we will see that our parameters give in output areas in the mountains, where only little villages exists.

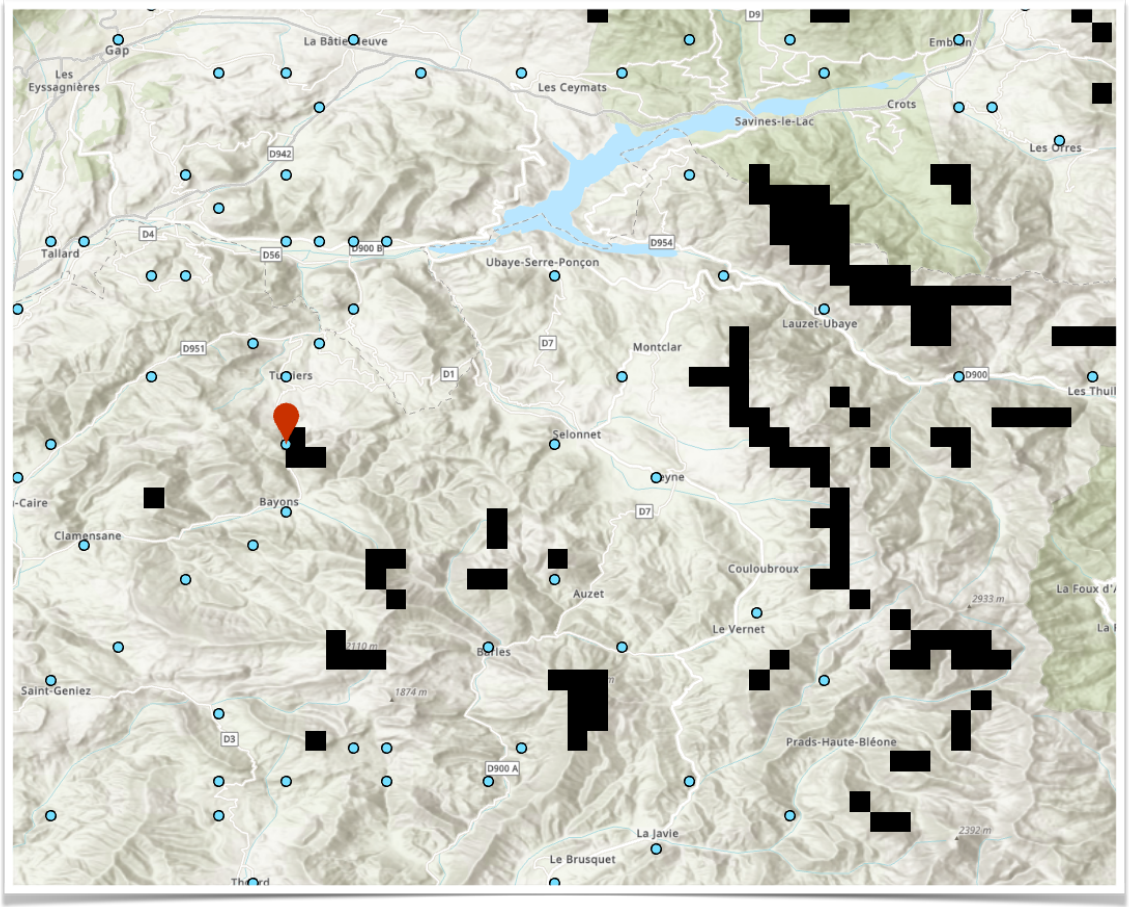
Moreover, the presence of a village can assure us that the slope is not too high to build an observatory.



Results

Simulation 1

```
coef1 = coef2 = coef3 = 1
citysize = 50,000 inhabitants
buffersize1= 30km
buffersize2= 30km
percentage of best pixels= 10%
```



Number of results:

Only 1 city in output.

FinalResult

Field: Add Calculate Selection: Zoom To Switch Clear Delete Copy

FID	Shape	FID_Bestpi	Id	gridcode	FID_french	Nom_Ville	MAJ	Code_Posta	Code_INSEE	Code_RŽgi	Latitude	Longitude
0	Point	99	100	1	1586	Astoin	ASTOIN	4250	4011	93	44.366667	6.166667

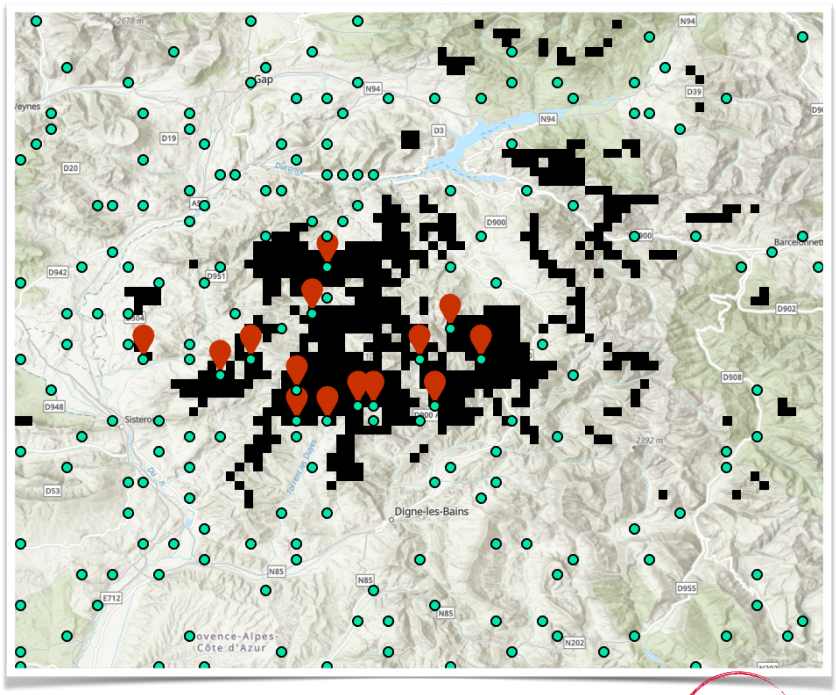
Results

Simulation 2 & 2.1

```
coef1 = 3
coef2 = 1
coef3 = 2
citysize = 50,000
buffersize1= 30km
buffersize2= 30km
```

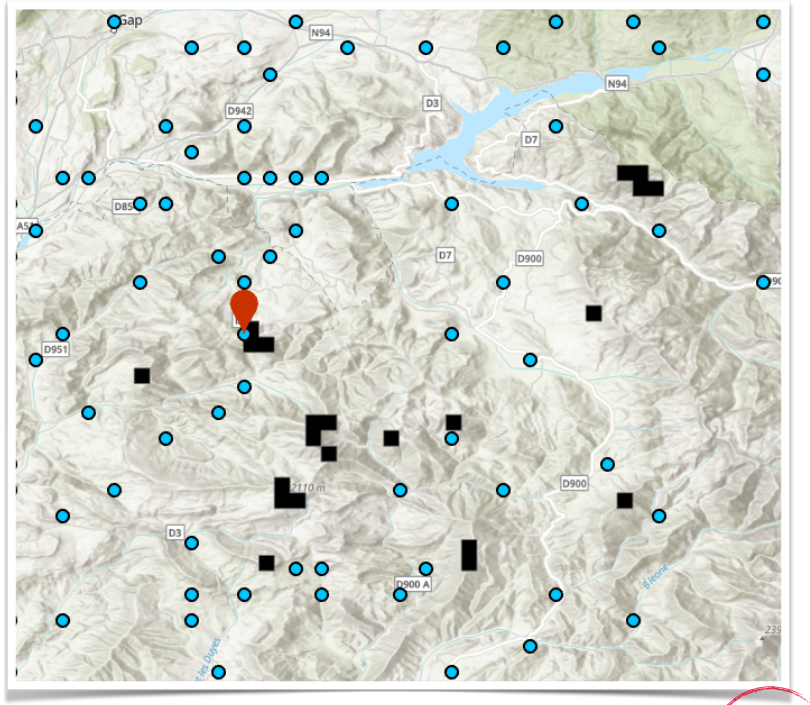
Luminous pollution is set as the most important criterion

```
Nbr of results:  1
Cities:
Name :  Astoin , GPS Coordinates:  ( latitude: 44.366667 longitude: 6.166667 )
```



14 outputs

1 output



`percentage of best pixels= 10%`

`percentage of best pixels= 5%`

Conclusion

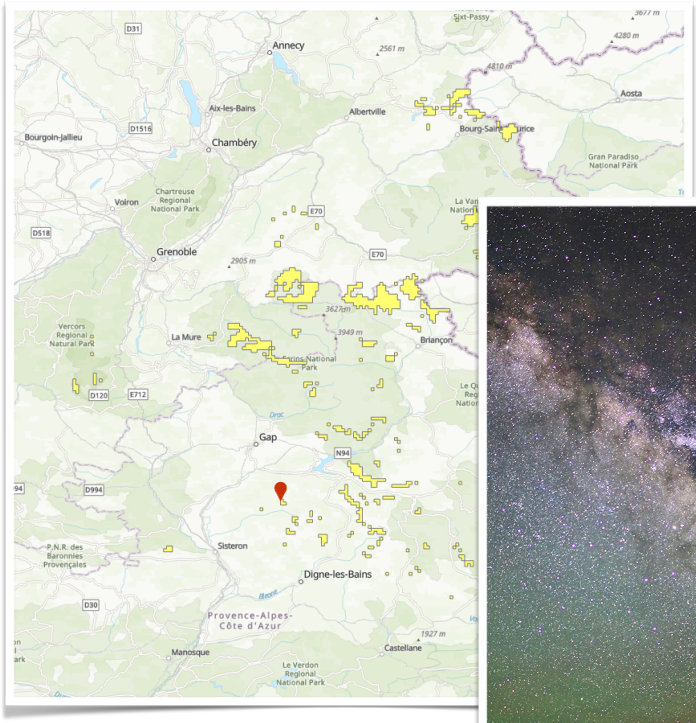
Technical conclusion:

By choosing the parameters of the function, one can create a whole new analysis based on their own criteria. The solution will be completely different but will respect the requests.

My simulations have gave in output approximately the same areas in the French Alps. Two little villages came out: Besse and Astoin.

Personal conclusion:

This project has been a great experience to me, and quite challenging. I have gained experience on ArGIS pro, and learned the usefulness of GIS, and coding with Python on Jupiter Notebook. Moreover, I learned how to use and push data into GitHub.
I would like to thank Naci Dilekli for this class.



EPF ÉCOLE D'INGÉNIEUR-E-S

RUDN university

```
def Find_Observatories
(workspace,fc1,b1,fc2,b2,citysize,
  coef1,coef2,coef3,percentage):
```

Credits

Luminous Pollution & Cloudiness:
Image and data processing by NOAA's National Geophysical Data Center. DMSP data collected by US Air Force Weather Agency.
https://ngdc.noaa.gov/eog/dmsp/downloadV4composites.html#AVSLCFC

Altitude:
Copernicus Land Monitoring Services
https://land.copernicus.eu/imagery-in-situ/eu-dem/eu-dem-v1.1?tab=mapview

GPS Coordinates of French cities:
https://www.galichon.com/codesgeo/index.php

Number of people in French cities in 2017:
INSEE
https://www.insee.fr/fr/statistiques/4265429?sommaire=4265511

List of French observatories:
Wikipedia https://fr.wikipedia.org/wiki/Liste_d%27observatoires_astronomiques
Google Maps (request: "Astronomical Observatories")

Thank you