MINES PARIS | PSL

Développement d’un indicateur d’accessibilité aux transports en commun

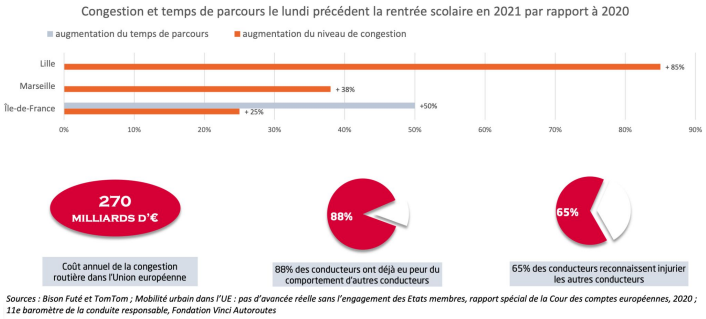
Auvergne
Rhône-Alpes
Énergie Environnement

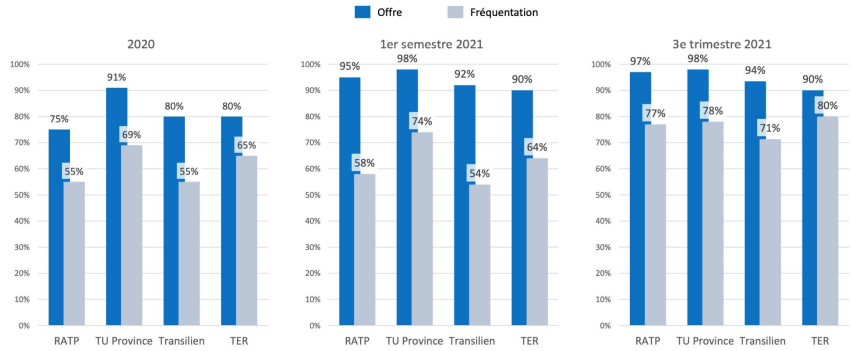
Contexte

- Étalement urbain
- Réchauffement climatique : 36% des GES émis par transports
- Urbanisation : diminuer la congestion

Solution :

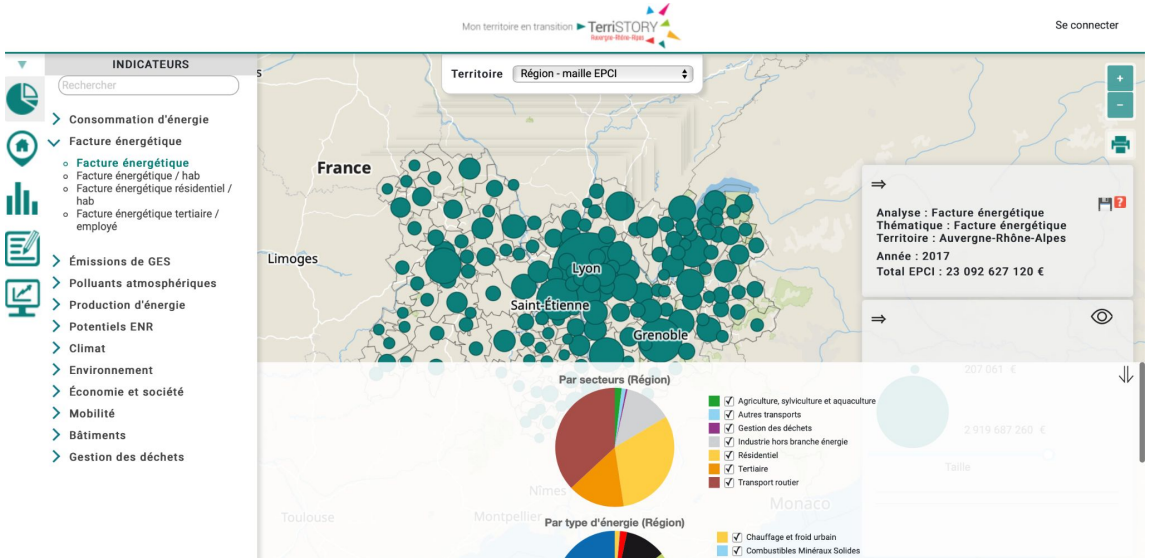
Augmenter l'utilisation des transports en commun en repensant la manière dont le réseau est construit





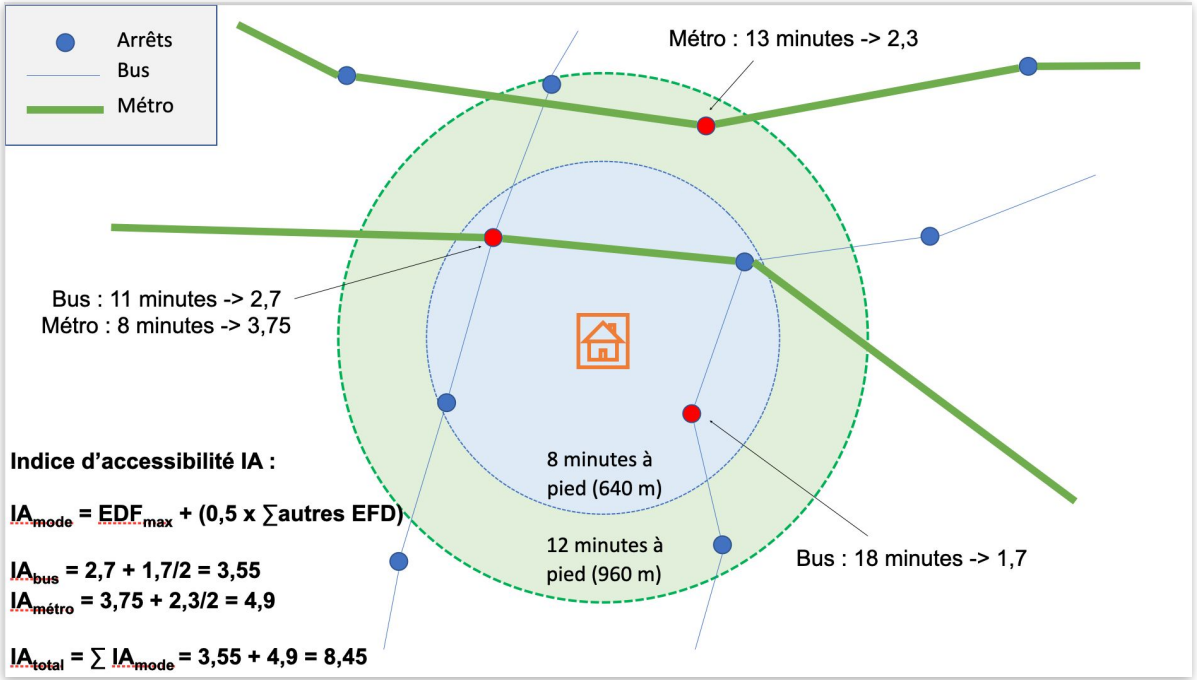
Présentation du projet

- Client : Auvergne-Rhône-Alpes Énergie Environnement
- But : créer un nouvel indicateur pour TerriSTORY : indicateur d'accessibilité aux transports en commun



Capture d'écran du site TerriSTORY

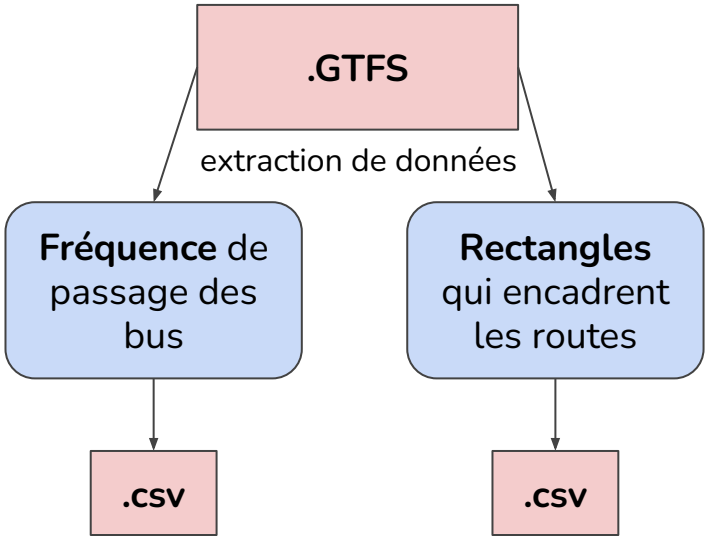
Calcul de l'indicateur



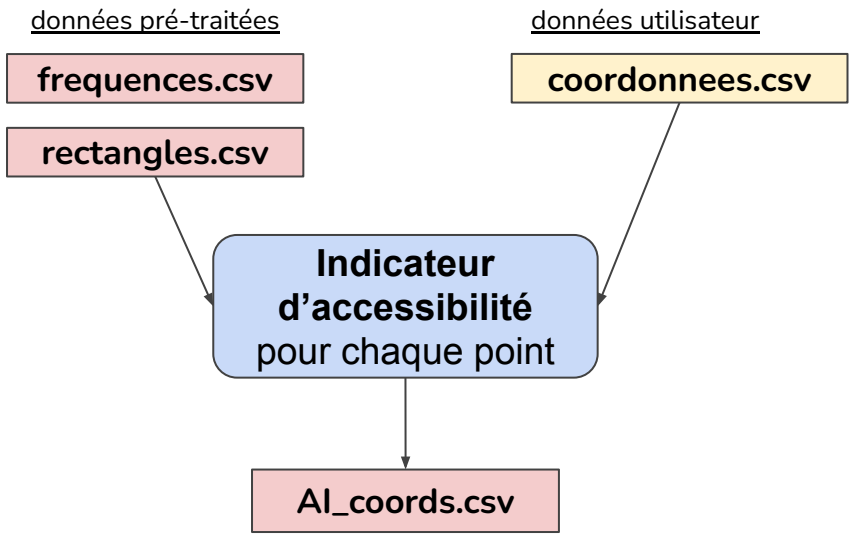
Support méthodologique fourni

Architecture globale du code

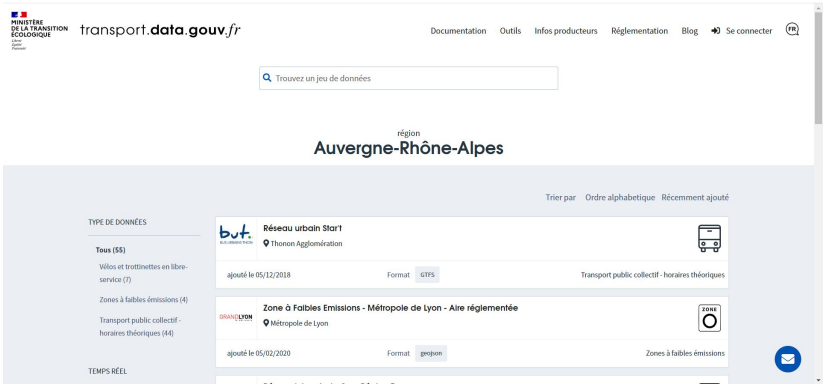
Traitement des données GTFS



Programmation de l'indicateur



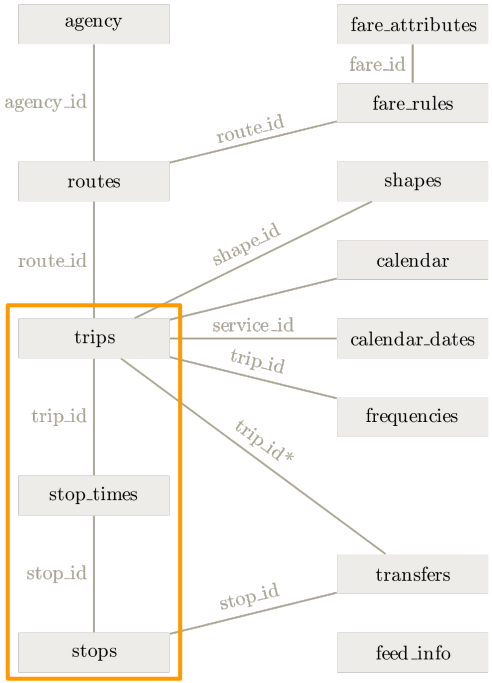
Récupération des données



Sur https://transport.data.gouv.fr/

Données accessibles gratuitement

Principe algorithme des rectangles



trips = un bus, un horaire
stops = position des arrêts
stop_times = heure de passage d'un bus à chaque arrêt

1) Choisir un lieu (un dossier .zip)

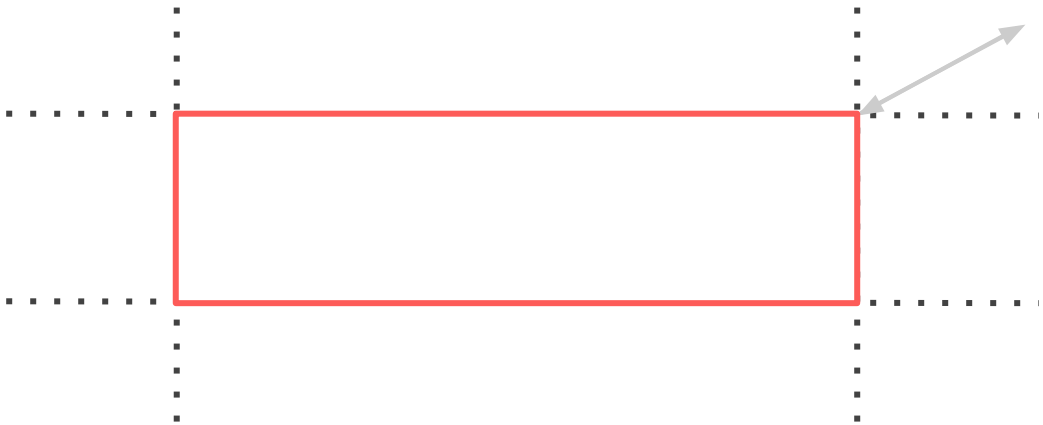
2) Choisir une route

3) Observer les trips associés

4) Accéder aux stops extrémaux du trip pour construire le rectangle

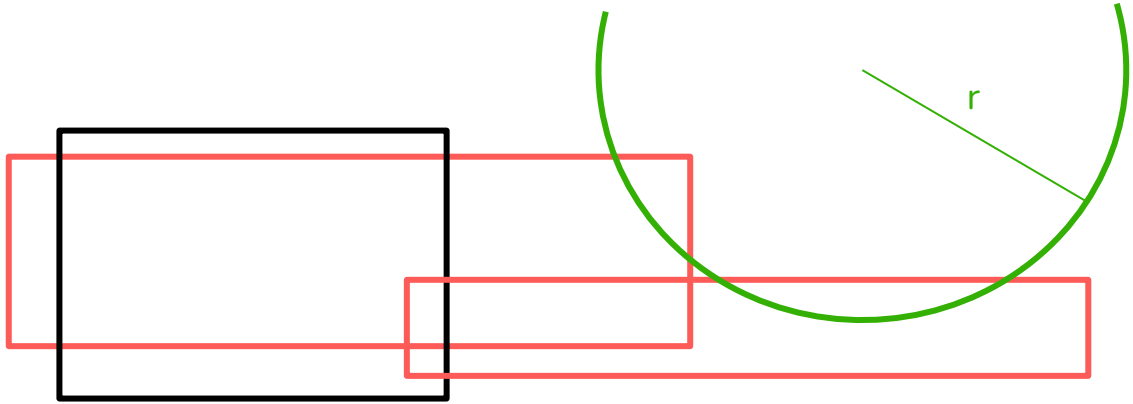
5) Conserver le plus grand rectangle pour la route choisie

Principe algorithme sélection rectangle

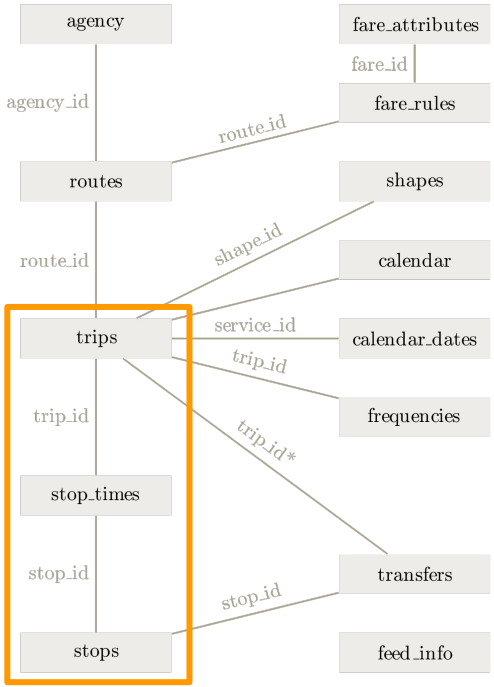


9 zones à différencier

Principe algorithme sélection rectangle



Principe algorithme fréquences



1) Prendre les DataFrame *trips* et *stop_times*

2) Regrouper par arrêt et route pour compter le nombre de bus par jour

3) Une journée approx. 12h → **nombre de bus par heure**

4) Rajouter la position des arrêts avec *stops*

Indicateur non-vectorisé

```
d = distance(lat, long, stop_lat, stop_long)
if d < marche_max: #si la distance à l'arrêt est plus pet
    temps_marche = (
        d / 75
    )  # on suppose que la vitesse moyenne d'un humain es
    temps_attente_moy = 0.5 * 60 / bus_per_hour
    temps_acces_tot = temps_attente_moy + temps_marche
    EDF = 30 / temps_acces_tot #ces calculs ci sont iss
    array_EDF = np.vstack((array_EDF, [route_id, EDF])) #
```

Pour chaque point de la carte du fichier d'entrée :

1) Choisir les rectangles
2) Parcourir tous les arrêts des rectangles retenus et **calculer l'EDF** si l'arrêt est assez proche
3) Stocker toutes les EDF dans une DataFrame
4) Eliminer les doublons (deux arrêts d'une même route)
5) Calculer l'EDF totale
6) Convertir en indicateur standardisé

Lyon : EDF=300
Clermont : EDF=40
Petite ville : EDF = 1

Cas d’utilisation

coords.csv

```
1   46.1510676,6.3494341
2   45.249431,4.672074
3   44.789, 3.584
4   45.188529, 5.724524
```

AI_coords.csv

```
1   lat,long,AI
2   46.1510676,6.3494341,0.8
3   45.249431,4.672074,2.0
4   44.789,3.584,0.8
5   45.764043,4.835659,6.2
```

PTAL	Range of Index	Map Colour	Description
1a (Low)	0.01 – 2.50		Very poor
1b	2.51 – 5.00		Very poor
2	5.01 – 10.00		Poor
3	10.01 – 15.00		Moderate
4	15.01 – 20.00		Good
5	20.01 – 25.00		Very Good
6a	25.01 – 40.00		Excellent
6b (High)	40.01 +		Excellent

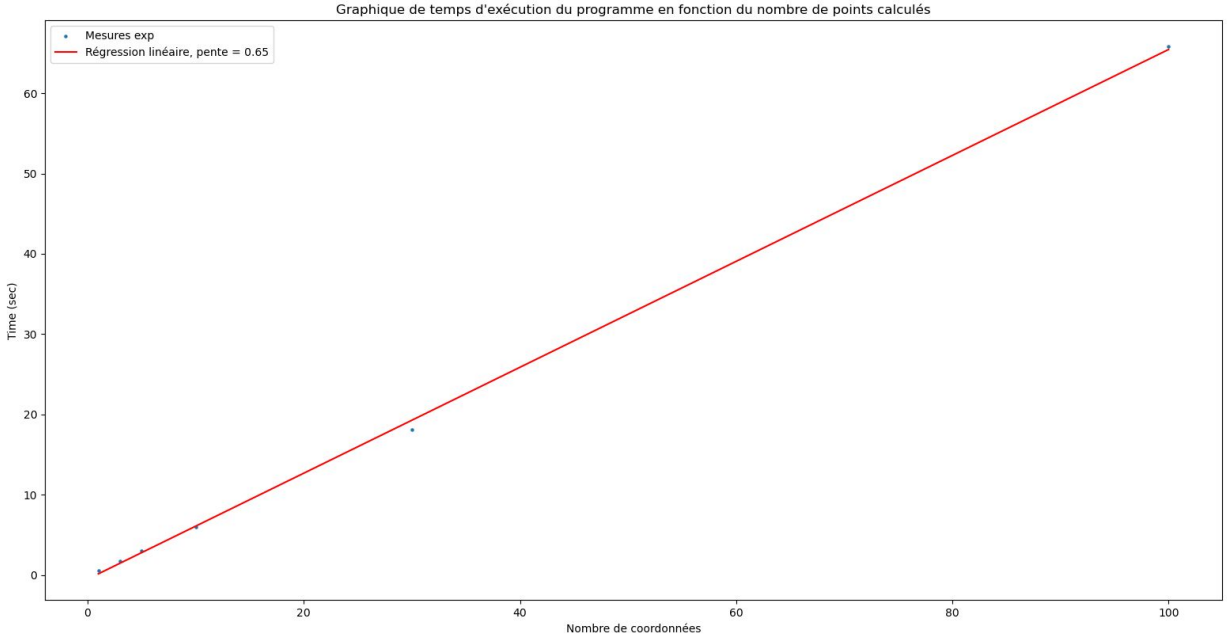
Mesuring PTALs (Public Transport Accessibility Level, Transport for London, 2010

Temps d’exécution et complexité

Mesure de temps d’exécution sur **différents nombres de points.** Plusieurs essais sur chaque nombre pour **moyenner.**

C = O(nombre de points)

Calcul d’un point ≈ 0,65 sec
→ Possibilité de **calcul en direct**



Indicateur vectorisé

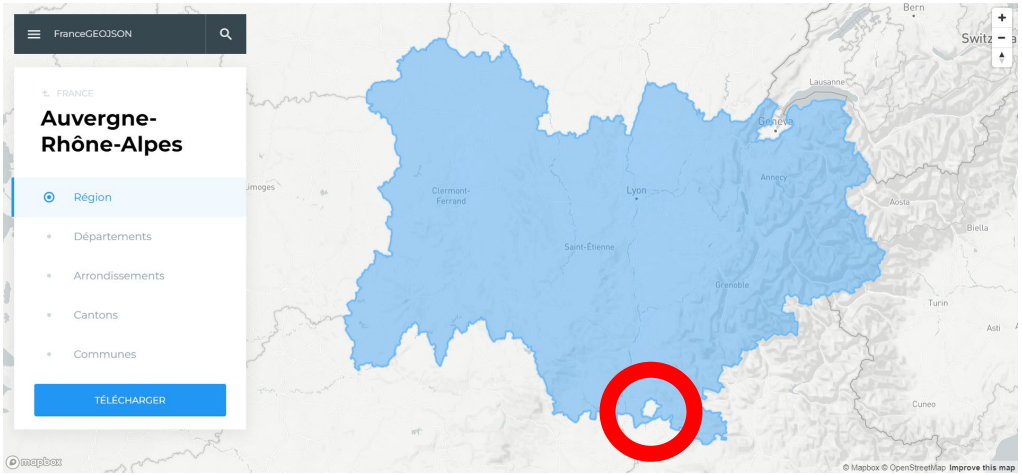
Indicateur non vectorisé → complexité linéaire → pas adapté à un grand nombre de points

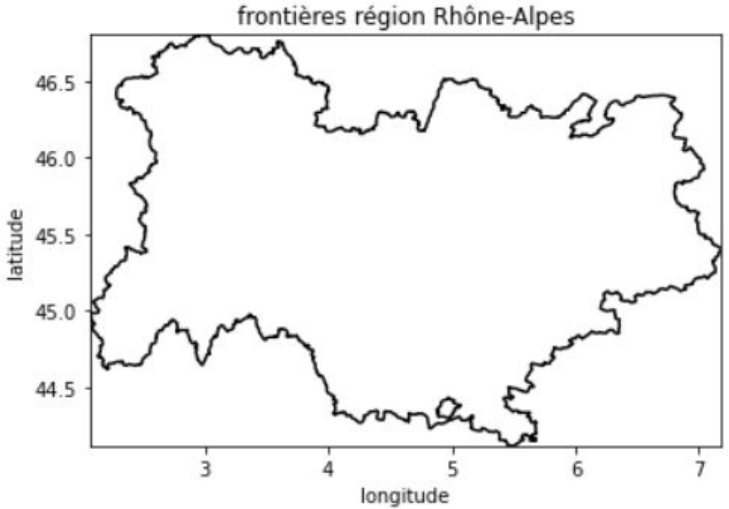
	Non-vectorisé	**Vectorisé**
Facteur limitant	Nombre d'arrêts	Nombre de points
Méthode	Parcourir tous les points Eviter les arrêts inutiles	Parcourir tous les arrêts Eviter les points inutiles

```
for ind in df_freq.index : #on parcourt les arrêts et on va distribuer les EDF aux cases du tableau environnantes
    agency_name, stop_id, route_id, bus_per_hour, stop_lat, stop_long = df_freq.iloc[ind]
    i, j = int((stop_long-long_min)/d_long), int((lat_max-stop_lat)/d_lat)
    for d_i in range(-i_close_cases, +i_close_cases+1) :
        for d_j in range(-j_close_cases, +j_close_cases+1) :
```

Cartographie

On obtient les coordonnées géographiques de la frontière de région Auvergne Rhône-Alpes sur https://france-geojson.gregoiredavid.fr/ au forma GeoJSON





https://france-geojson.gregoiredavid.fr/

Réalisation d'un masque

```
plt.fill(contour_region[:,0], contour_region[:,1], c='black')
plt.fill(contour_enclave[:,0], contour_enclave[:,1], c='white')

plt.xlim([longitude_min, longitude_max])
plt.ylim([latitude_min, latitude_max])

plt.axis('off')

plt.savefig('out.jpg', dpi= 200, bbox_inches = None)
```

✓ 0.2s

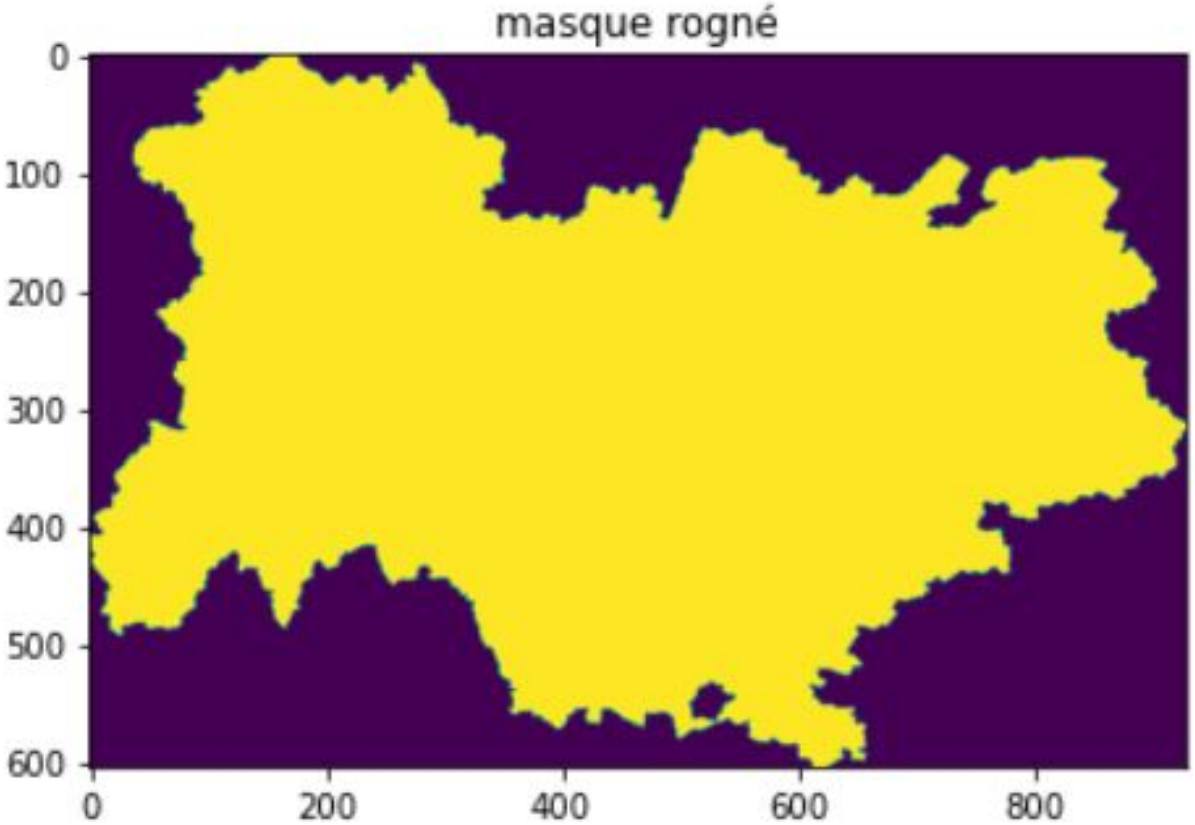
```
def test_ligne(a):
    only_false = True
    i=0
    while only_false == True and i<len(a):
        if a[i] == True:
            only_false = False
        i+=1
    return only_false

for j in range(im_mask.shape[0]):
    if test_ligne(im_mask[j]):
        compt_ligne+=1
```

✓ 1.2s

```
a = np.zeros((im_mask.shape[0]-compt_ligne, im_mask.shape[1]))
i = 0
j = 0
for i in range(im_mask.shape[0]):
    if not(test_ligne(im_mask[i])):
        a[j] = im_mask[i]
        j+=1

plt.imshow(a)
```



-> Taille de l’image connue, dépend du paramètre dpi et du nombre de lignes/colonnes vides qui sont supprimées.

```python
latitudes = np.linspace(latitude_min, latitude_max, nb_lignes)
longitudes = np.linspace(longitude_min, longitude_max, nb_col)

coords = np.zeros((nb_lignes, nb_col, 2))

for i in range(nb_lignes):
    for j in range(nb_col):
        coords[i,j,0] = longitudes[j]
        coords[i,j,1] = latitudes[-i-1]

map_finale = np.concatenate((new_im_crop,coords), axis=2)
```

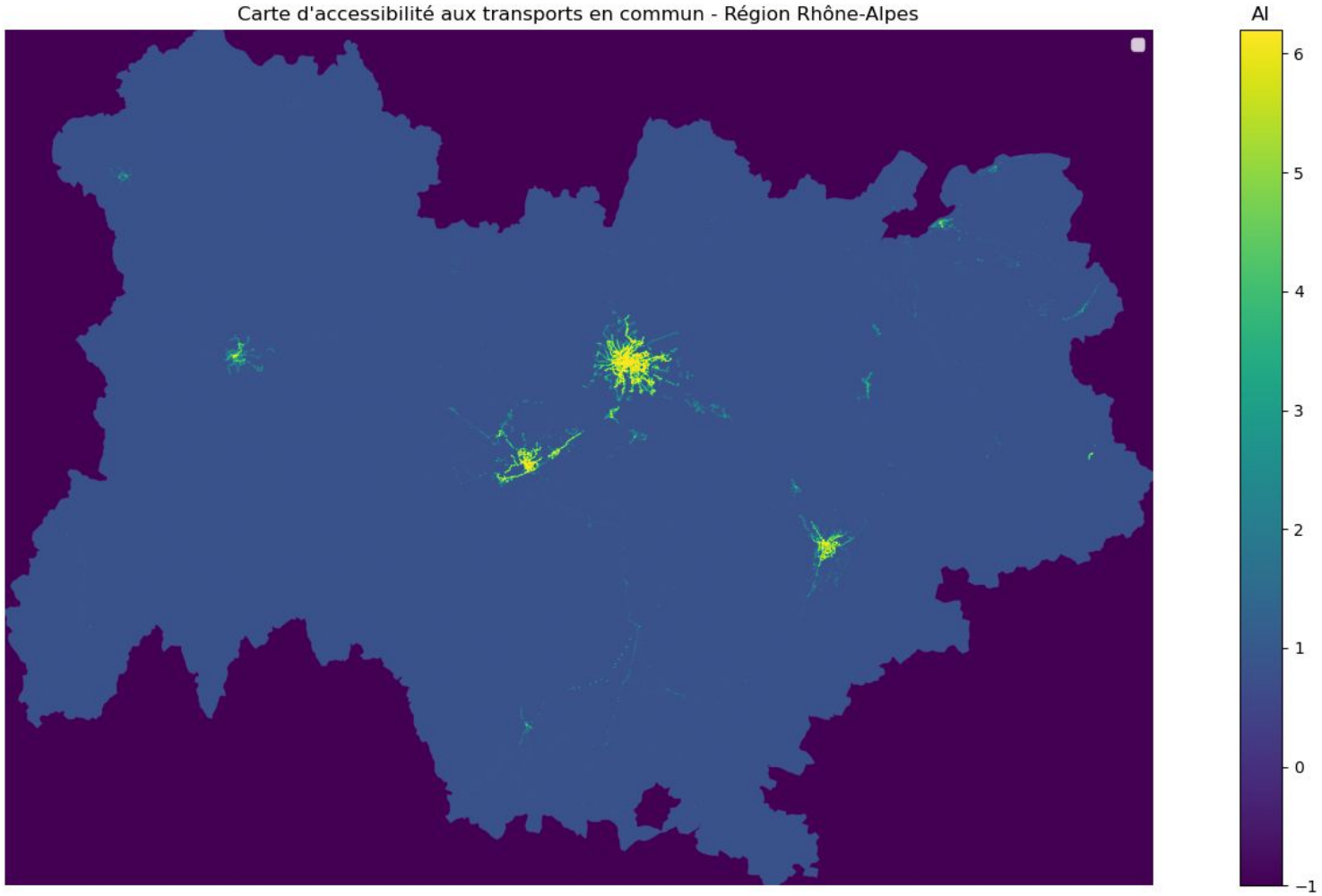
```python
print(map_finale.shape)
map_finale[100,100,:]
```

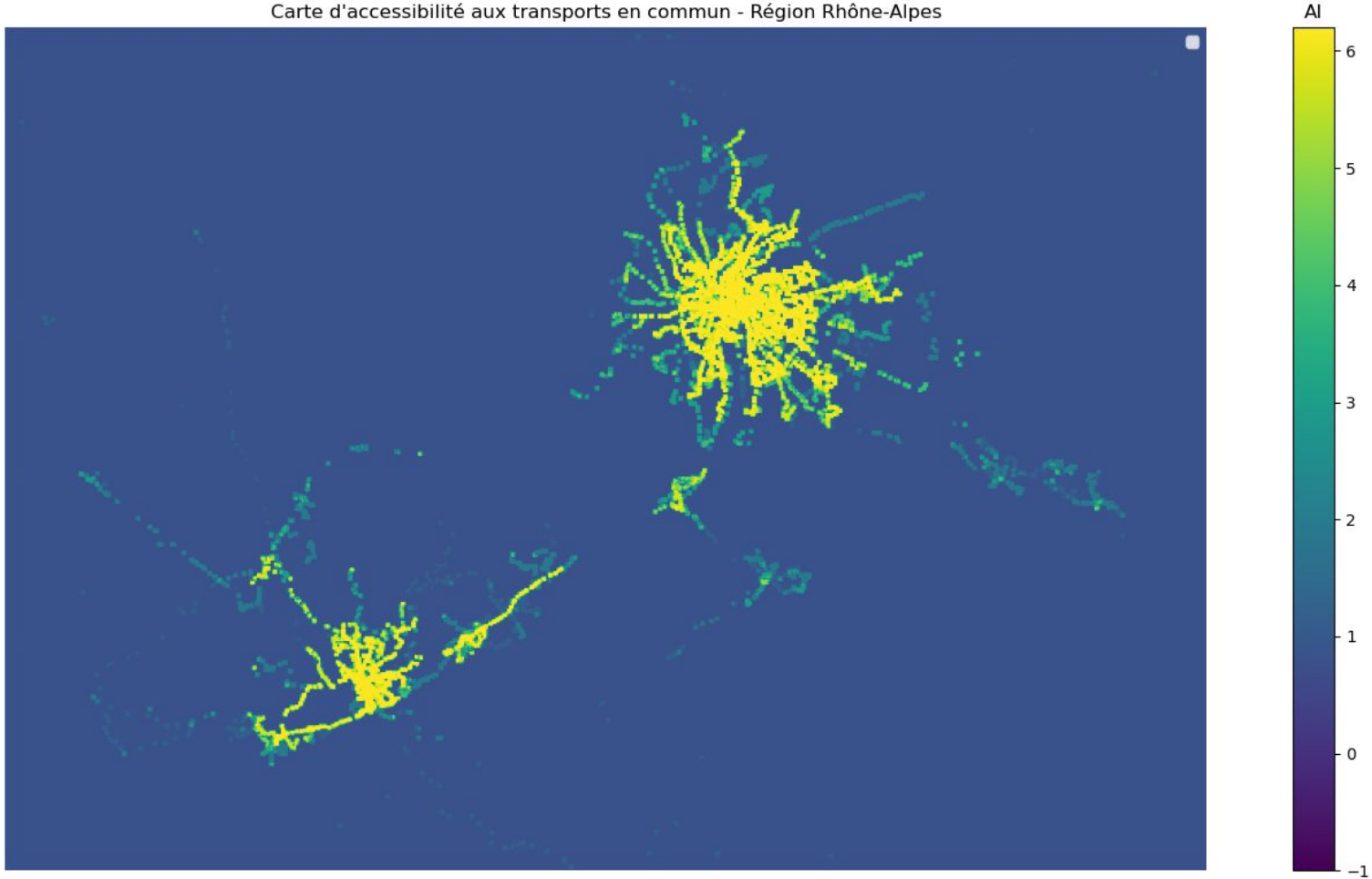
✓ 0.9s

```
(604, 930, 3)

array([ 1.        ,  2.61418497, 46.35807516])
```

Ce point est dans la région, longitude 2.614 et latitude 46.358


Carte d'accessibilité aux transports en commun - Région Rhône-Alpes
AI
6
5
4
3
2
1
0
−1


Carte d'accessibilité aux transports en commun - Région Rhône-Alpes
AI
6
5
4
3
2
1
0
−1

Quelques limites et améliorations envisageables

- Si on veut étendre : RER A approx. autant de données que la région Rhône-Alpes → temps de création des fichiers .csv d'origine **très long**.

- Les données ne sont **pas centralisées** (chaque compagnie publie ses données), il faut donc toujours surveiller que les données de toutes les compagnies sont **à jour**.

- Complexifier la fonction vectorisée pour qu'elle prenne un **polygone** et non un tableau rectangulaire en entrée.

- **Hyper-threading**.

Conclusion

- Travailler avec un **client**, construire un projet brique par brique en y apportant des modifications

- **Adapter des données au format particulier** à une utilisation précise

- Apprendre à **optimiser** et à travailler **en parallèle**

- Garder un code **propre**