

Autoroutes du Sud-Ouest de la France

Groupe 7

ABDERRAHIM AIT MOULAY
ADRIEN SIMON
ASSANE SENE

FDS Montpellier

Le 13 Décembre 2021

Plan

- ① Introduction
- ② Nettoyage de données
- ③ La carte interactive
- ④ Programme d'optimisation
- ⑤ KDE de la distribution des prix par kilomètre
- ⑥ Conclusion

Répartition du travail

- Abderrahim s'est occupé de la carte interactive, du nettoyage de la base de données des distances et de la mise en place de la documentation
- Assane s'est occupé de la répartition des prix et du nettoyage de la base de donnée des prix
- Adrien s'est occupé de la minimisation du coût du trajet et du calcul des distances à partir des coordonnées GPS

Nettoyage de données

Creation dataframe et matrice prix

- Importation des données avec la fonction **pd.read_csv()**:

```
df_p = pd.read_csv("Data_prix.csv").
```

- Remplacement des NAN par 0.0 dans le dataframe:

```
df_p = df_p.fillna(0.0).
```

- suppression des colonnes (ou des sorties) où l'on ne peut pas sortir:

Avec la fonction **del**, nous allons supprimer les sorties auxquelles on ne peut pas sortir. Ces dernières sont:

- Peage de Montpellier St-Jean
- Peage du Perthus
- Le Boulou (peage sys ouvert)
- Peage de pamiers
- Peage de Toulouse sud/ouest
- Peage de Toulouse sud/est

Nettoyage de données

- remplacement de 0 par 0.0 et 8.58,5 par 8.5 dans les colonnes Vendargues et Narbonne sud du dataframe et affichage du nouveau dataframe:
- transformation du dataframe df_p en matrice (numpy array):

```
df_pnmp = np.array(df_p, dtype = 'float64')
```

La carte interactive

Objectif

L'objectif principal de cette partie est de créer une carte interactive avec le package [folium](#) afin d'afficher l'itinéraire, la distance, le prix ainsi que le prix par kilomètre entre deux gares données par l'utilisateur.

La carte interactive

Objectif

L'objectif principal de cette partie est de créer une carte interactive avec le package [folium](#) afin d'afficher l'itinéraire, la distance, le prix ainsi que le prix par kilomètre entre deux gares données par l'utilisateur.

Fonction

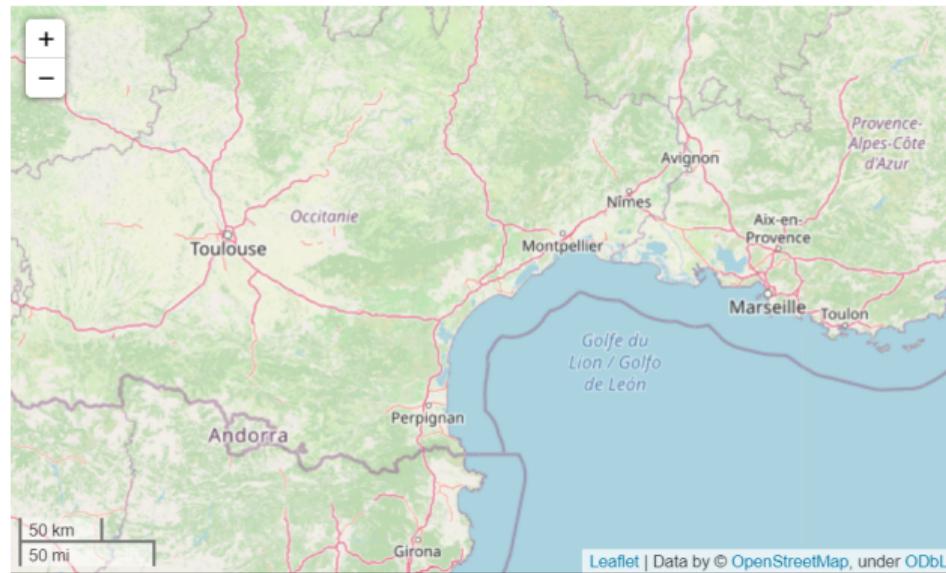
```
def Distab(a,b):  
    return (Dist.iloc[a][b+1])
```

```
def prixab(a,b):  
    return (prix.iloc[a][b+1])
```

La carte interactive

Nous avons utiliser la fonction "folium.map" qui nous permet d'afficher une carte vide.

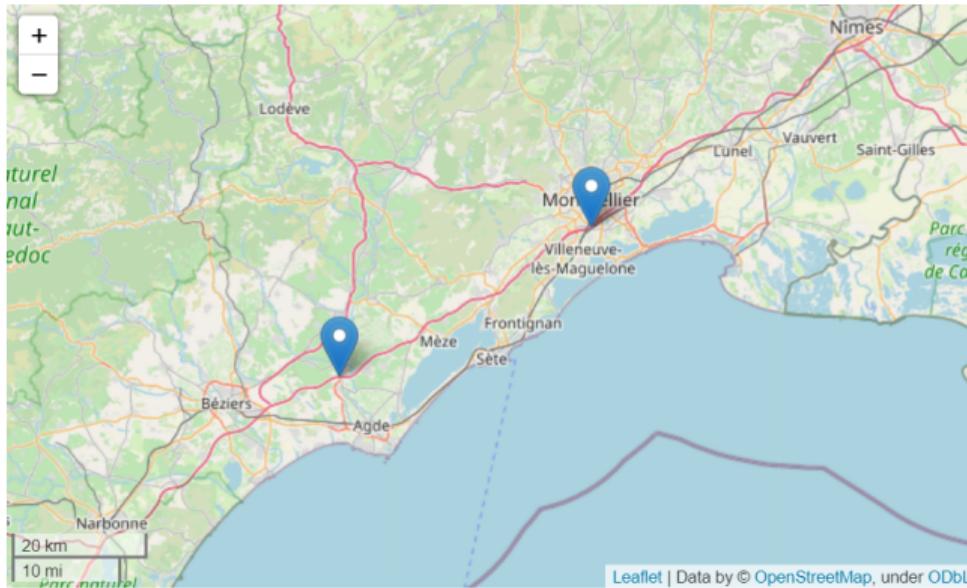
```
folium.Map(location=.....,zoom_start=...., control_scale=....)
```



La carte interactive

Pour afficher un point sur la carte nous avons utiliser la fonction "folium.Marker".

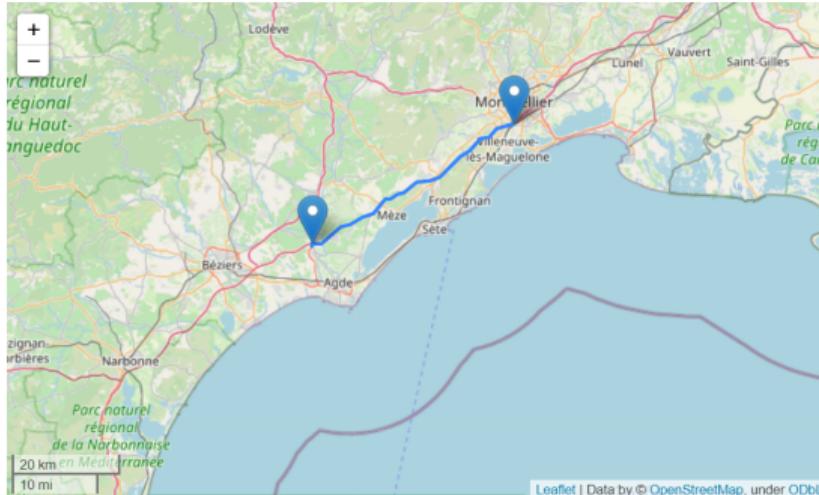
```
folium.Marker( location=list(.....), ).add_to(m)
```



La carte interactive

Pour tracer la route entre des points donnés nous avons utiliser la fonction "folium.GeoJson".

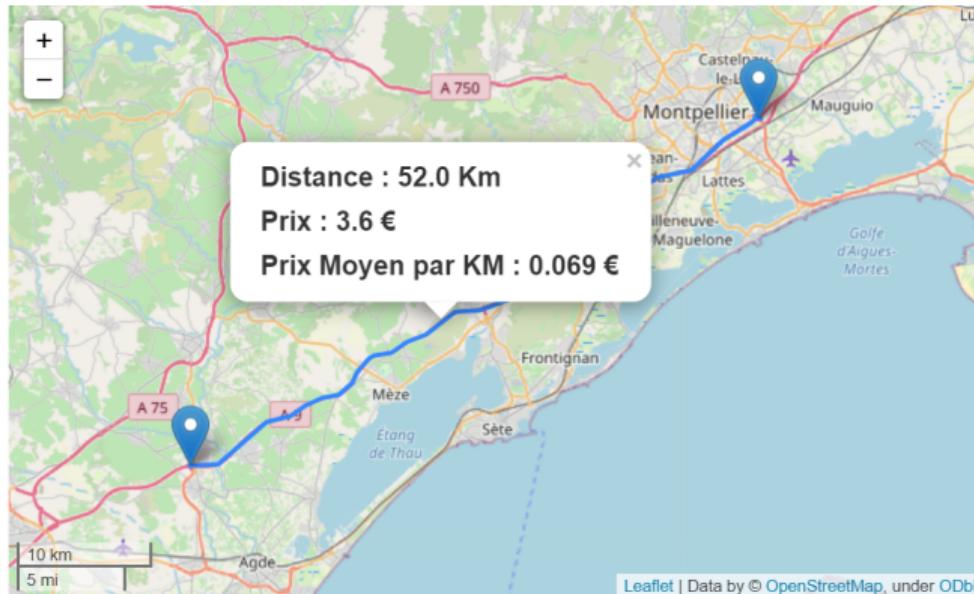
```
geometry = client.directions(coords)['routes'][0]['geometry'] decoded =  
convert.decode_polyline(geometry)  
folium.GeoJson(decoded).add_child(folium.Popup(max_width=300)).add_to(m)
```



La carte interactive

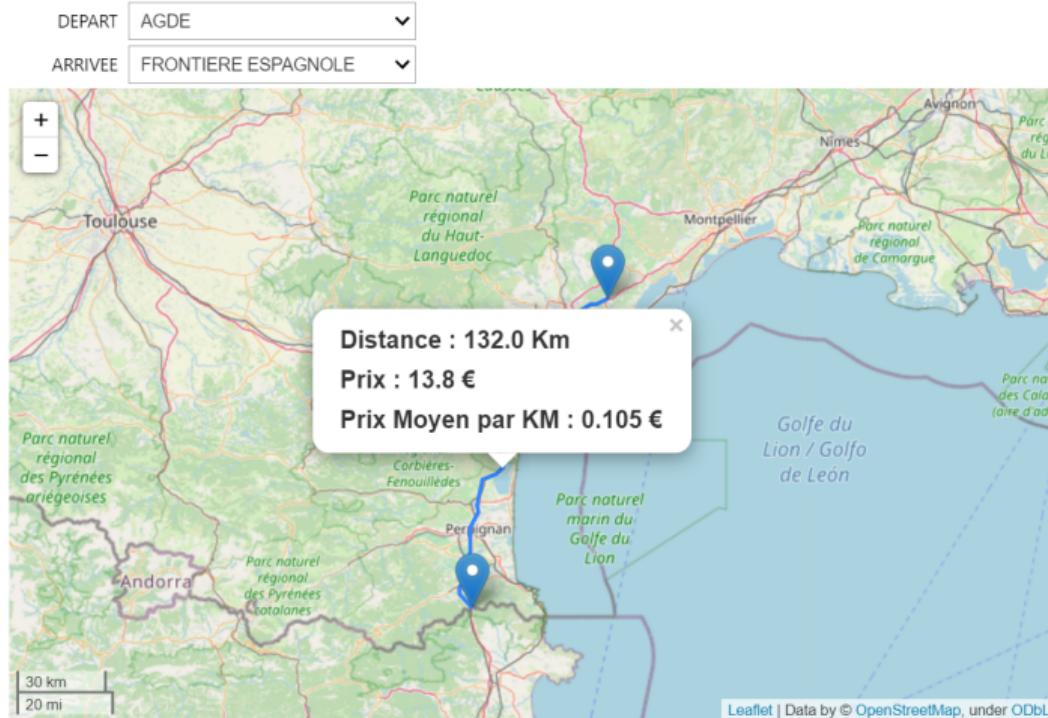
Pour ajouter le texte de la fenêtre contextuelle lorsque nous cliquons sur l'itinéraire.

```
folium.GeoJson(decoded).add_child(folium.Popup(texte,max_width = 300)).add_to(m)
```



Widgets

interact(itineraire, DEPART = villes, ARRIVEE = villes)



Programme d'optimisation

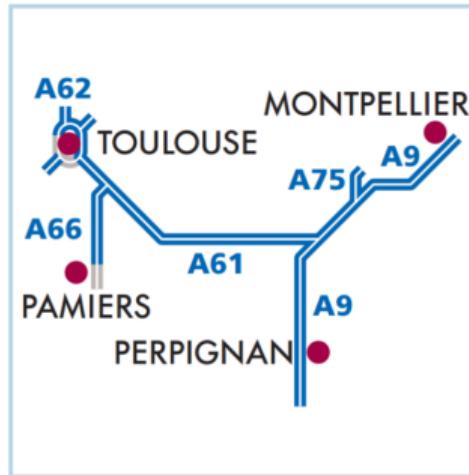
Rappel de l'objectif

Le but de cette partie est de minimiser le coût du trajet selon un nombre de sorties intermédiaires maximum donné.

Etapes intermédiaires

- Comment ordonner les sorties ?
- Comment faire pour obtenir toutes les combinaisons de n sorties entre deux points sur l'autoroute ?
- Comment traiter les sorties "problématiques" ?
- Comment fonctionne l'algorithme et comment pourrait-on l'optimiser ?

Passer d'un système sans ordre à un système ordonné



Portion 1	Vendargues	...	Narbonne Sud
Portion 2	Sigean	...	Frontière Espagnole
Portion 3	Lezignan	...	Villefranche-de-Lauragais
Portion 4	Nailloux	...	Pamiers Sud
Portion 5	Montgiscard	...	Sesquières

Utilisation de itertools

Fonction itertools.combinations

J'ai combiné la fonction combinations de la librairie itertools avec ma fonction nbSorties pour obtenir la fonction que j'ai nommée march()

```
march(4,15,3)
[0]    ✓  0.1s
... [[4, 5, 6, 7, 15],
     [4, 5, 6, 8, 15],
     [4, 5, 6, 9, 15],
     [4, 5, 6, 10, 15],
     [4, 5, 6, 11, 15],
     [4, 5, 6, 12, 15],
     [4, 5, 6, 13, 15],
     [4, 5, 6, 14, 15],
     [4, 5, 7, 8, 15],
```

Sorties à problèmes

traiter les cas particuliers

Pourquoi as-t-on décidé de ne pas prendre en compte les péages ?

Comment faire pour contourner les problèmes posés par les sorties Le boulou sur la A9 et Pamiers Nord sur la A66 ?

Fonctionnement global et optimisation du code

fonctionnement de base

Combiner la fonction Finale() et l'utilisation de widgets

DEPART	Montpellier est
ARRIVEE	Montgiscard
Nombre	5

Pour aller de Montpellier est à Montgiscard
Nous vous conseillons les sorties suivantes :
['Sete', 'Agde Pezenas', 'Beziers ouest', 'Lezignan', 'Carcassone est']
Au lieu de payer 21.2€ vous aurez payé 17.3€.
Vous économisez 3.9€.

Fonctionnement global et optimisation du code (suite)

NetworkX et graphs

Autres méthodes pour la partie de minimisation du prix du trajet : Les graphs et l'algorithme de Kruskal

Optimisation du code

Comment pourrait-on réduire les temps de calcul en conservant notre méthode de résolution de la problématique ?

KDE de la distribution des prix par kilomètre

Objectif

Dans cette partie, nous allons essayer de donner un KDE (distribution, estimation, tracer de courbe, ...) des prix par km entre une entrée et une sortie choisies, à l'aide de fonction interact contenue dans le module ipywidgets. Dans un premier temps, nous allons faire un nettoyage des données. Puis , nous créerons un data frame des prix par km. Enfin,nous ferons un KDE de ce data frame obtenu.

KDE de la distribution des prix par kilomètre

Création des fonctions

- def Distribution_Prix(i,j):

Cette fonction calcule le prix entre toutes les sorties successives parmi toutes les sorties qui se trouvent entre i et j.

- def Distribution_dist(i,j):

Cette fonction calcule la distance entre toutes les sorties successives parmi toutes les sorties qui se trouvent entre i et j.

KDE de la distribution des prix par kilomètre

Fonctions

- def moyDist(i,j):

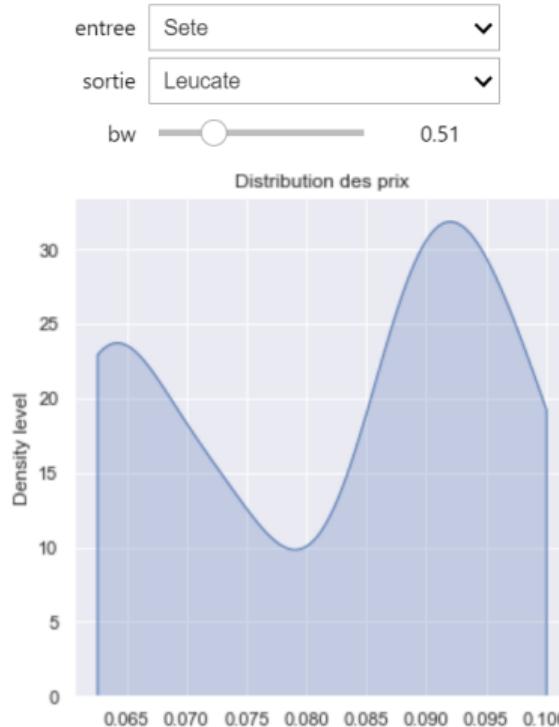
Cette fonction permet de calculer la prix par km entre toutes les sorties successives parmi toutes les sorties qui se trouvent entre i et j.

- creation du vecteur villes:

```
villes = []...
```

KDE de la distribution des prix par kilomètre

```
interact(kde_explore, entre = villes, sortie = villes, bw = (0.001, 2, 0.01))
```



Conclusion

le programme asof.py

Nous avons finalement combiné l'ensemble de notre travail dans un seul fichier python qui appelle les trois parties de ce projet et qui permet d'avoir l'affichage de la carte interactive, le trajet le moins cher sur une section d'autoroute et la répartition des prix. Il suffit d'executer le fichier asof.py et de choisir les sorties qui vous intéressent dans les widgets interactifs !



MERCI POUR VOTRE ATTENTION !!