

Projet python

Analyse du réseau SNCF

Diego Renaud, Victor Parent, Marion Chabrol



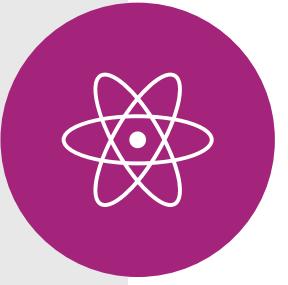
SOMMAIRE



Récupération et traitement des données



Analyse descriptives et représentation graphique



Modélisation



Présentation du mini-jeu

Récupération des données

1. API SNCF: Catalogue

Catalog API to enumerate datasets

GET /catalog/datasets Query catalog datasets

GET /catalog/exports/{format} Export a catalog

GET /catalog/facets List facet values

```
df_catalogue = importer("https://ressources.data.sncf.com/api/v2/catalog/exports/json?limit=-1&offset=0&timezone=UTC")
pd.DataFrame(df_catalogue['dataset_id']).head()
```

	dataset_id
0	signalisation-permanente
1	rapports-irc
2	mode-de-cantonnement-des-lignes

Récupération des données

2. Sélection des Datasets utiles

```
#importation de la liste des gares, API SNCF (peut prendre une quinzaine de secondes)
df_gares = importer("https://ressources.data.sncf.com/api/v2/catalog/datasets/liste-des-gares/exports/json?limit=-1")
df_gares.head(2)
```

	code_uic	libelle	fret	voyageurs	code_ligne	rg_troncon	pk	commune	departemen	idreseau	idgaia
0	87755223	Bandol	N	O	930000	1 050+607	BANDOL	VAR	3077	29a5a626-dfbc-11e3-a2ff-01a464e0362d	

```
#importation des données sur la fréquentation des gares, API SNCF
df_freq_gares = importer("https://ressources.data.sncf.com/api/v2/catalog/datasets/frequentation-gares/exports/json")
df_freq_gares.head(2)
```

	nom_gare	code_uic_complet	code_postal	segmentation_drg	total_voyageurs_2021	total_voyageurs_non_voyageurs_2021	tc
0	Abbaretz	87481614	44170	c	27466	27466	

```
df_retards = importer('https://ressources.data.sncf.com/api/v2/catalog/datasets/regularite-mensuelle-tgv-aqst/expor
df_retards.head(2)
```

	date	service	gare_depart	gare_arrivee	duree_moyenne	nb_train_prevu	nb_annulation	commentaire_annulatio
0	2018-01	National	BREST	PARIS MONTPARNASSE	225	284	3	Nor

Traitement des données

1. Fusion des dataframes

```
#On regarde quelles gares du dataframe "liste des gares" ne sont pas dans le dataframe "frequentations"
outer_merged = pd.merge(df_gares, df_freq_gares, how="outer", on=["code_uic"])
outer_merged[outer_merged['nom_gare'].isna() == True][['code_uic', 'libelle']]
```

	code_uic	libelle
3	87784488	Cases-de-Pène
12	87396671	La Chartre-sur-le-Loir
35	87271320	La Courneuve-Dugny
39	87172411	Margut-Fromy
49	87172593	Vouziers
...
3785	87286807	Genech
3793	87755835	La Motte-Ste-Rosseline
3799	87729210	Port-Fluvial-de-Chalon-sur-Saône
3801	87286252	Rougebarre
3812	87182154	Huningue

551 rows × 2 columns

```
#Fusion des dataframes sur la base de df_freq_gares avec un left merge
left_merged = pd.merge(df_freq_gares, df_gares, how="left", on=["code_uic"])
df_gares_merged = left_merged[['code_uic', 'total_voyageurs_non_voyageurs_2020', 'nom_gare', 'x_wgs84', 'y_wgs84']]
df_gares_merged.head(2)
```

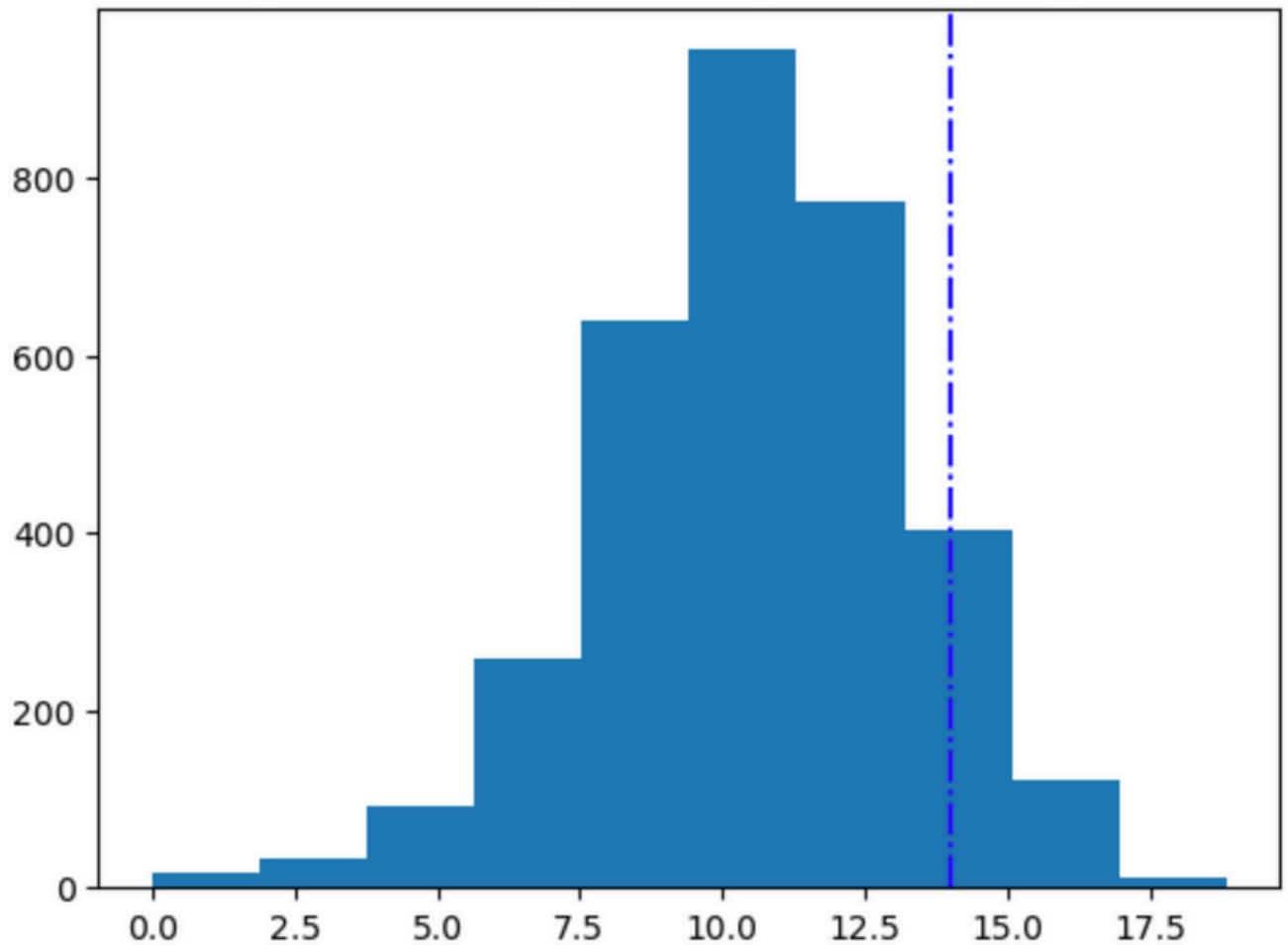
	code_uic	total_voyageurs_non_voyageurs_2020	nom_gare	x_wgs84	y_wgs84
0	87481614	22773	Abbaretz	-1.524313	47.555462
1	87271460	2177954	Aéroport Charles de Gaulle 1	2.555964	49.008827

Traitement des données

2. Filtrage

```
df_gares_merged['log_total_voyageurs_non_voyageurs_2020'] = np.log(df_gares_merged['total_voyageurs_non_voyageurs_2020'])
plt.hist(df_gares_merged['log_total_voyageurs_non_voyageurs_2020'])
plt.title("Répartition du log de la fréquentation des gares")
plt.axvline(x = 14, color = 'b', linestyle = '-.')
plt.show()
```

Répartition du log de la fréquentation des gares



```
#Application d'un filtre sur les frequentations : on ne garde que les gares les plus frequentées (à droite de la droite verticale)
df_gares_principales = df_gares_merged[df_gares_merged['log_total_voyageurs_non_voyageurs_2020'] >= 14]
```

Traitement des données

3. Recherche des données manquantes

	code_uic	total_voyageurs_non_voyageurs_2020	nom_gare	x_wgs84	y_wgs84	log_total_voyageurs_non_voyageurs_2020
384	87756403	4085871	Monaco-Monte-Carlo	NaN	NaN	15.223045
664	87113795	3664906	Villiers-sur-Marne - Le Plessis-Trévise	NaN	NaN	15.114313
1195	87113704	2380920	Rosny-sous-Bois	NaN	NaN	14.682998
1488	87758607	7980634	Châtelet les Halles	NaN	NaN	15.892528
1533	87116046	2496908	Émerainville - Pontault-Combault	NaN	NaN	14.730564
1806	87113209	3862190	Pantin	NaN	NaN	15.166745
1808	87758904	107666015	Paris Gare du Nord	NaN	NaN	18.494545
1852	87654798	2865667	Rosa Parks	NaN	NaN	14.868312
1855	87113696	3206275	Rosny Bois Perrier	NaN	NaN	14.980620
2347	87113522	1562549	Le Chénay Gagny	NaN	NaN	14.261829
2565	87116038	1913470	Roissy-en-Brie	NaN	NaN	14.464429
3023	87113779	1240481	Les Boulleaux Champigny	NaN	NaN	14.031010
3132	87113746	2091668	Nogent - Le Perreux	NaN	NaN	14.553472
3146	87113001	27242868	Paris Est	NaN	NaN	17.120302
3294	87113712	17692029	Val de Fontenay	NaN	NaN	16.688625

- **Problème** : il nous manque les données GPS de gares importantes (la gare de Paris EST par exemple)
- **Solution** : ces gares étant -presque- toutes situées en région Ile de France (idf) on va aller chercher ces données sur l'API de la région idf.

#Mise en évidence de l'erreur

```
print(df_region_idf_initial[df_region_idf_initial['libelle']=="PARIS NORD"][['libelle', 'code_uic']])
print(df_gares_manquantes[df_gares_manquantes['nom_gare']=="Paris Gare du Nord"][['nom_gare', 'code_uic']])
```

	libelle	code_uic
	nom_gare	code_uic
92	PARIS NORD	87271031
1808	Paris Gare du Nord	87758904

Traitement des données

3. Recherche des données manquantes

	code_uic	uic7	libelle_point_d_arret	libelle	libelle_stif_info_voyageurs	code_insee_commune	commune	x_lambert_ii_etendu	y_lambert_ii_etendu	coord_gps_wgs84
0	87113209	871132	PANTIN	PANTIN	GARE DE PANTIN	93055	[Pantin]	604688.0	2433279.0	{"lon": 2.40040674765, "lat": 48.8977738494}
1	87271031	872710	PARIS NORD (GARE DU NORD)	PARIS NORD	GARE DU NORD SURFACE	75110	[Paris]	601529.0	2431239.0	{"lon": 2.35733955798, "lat": 48.8794550008}
2	87116038	871160	ROISSY EN BRIE	ROISSY EN BRIE	GARE DE ROISSY EN BRIE	77390	[Roissy-en-Brie]	623033.0	2421933.0	{"lon": 2.64984936854, "lat": 48.7954076773}
3	87113696	871136	ROSNY BOIS PERRIER	ROSNY BOIS PERRIER	GARE DE ROSNY BOIS PERRIER	93064	[Rosny-sous-Bois]	610640.0	2431610.0	{"lon": 2.48149612336, "lat": 48.8827036731}
4	87654798	876547	Rosa Parks	Rosa Parks	GARE DE ROSA PARKS	75119	[Paris]	602710.0	2433136.0	{"lon": 2.37345, "lat": 48.8965}

Traitement des données

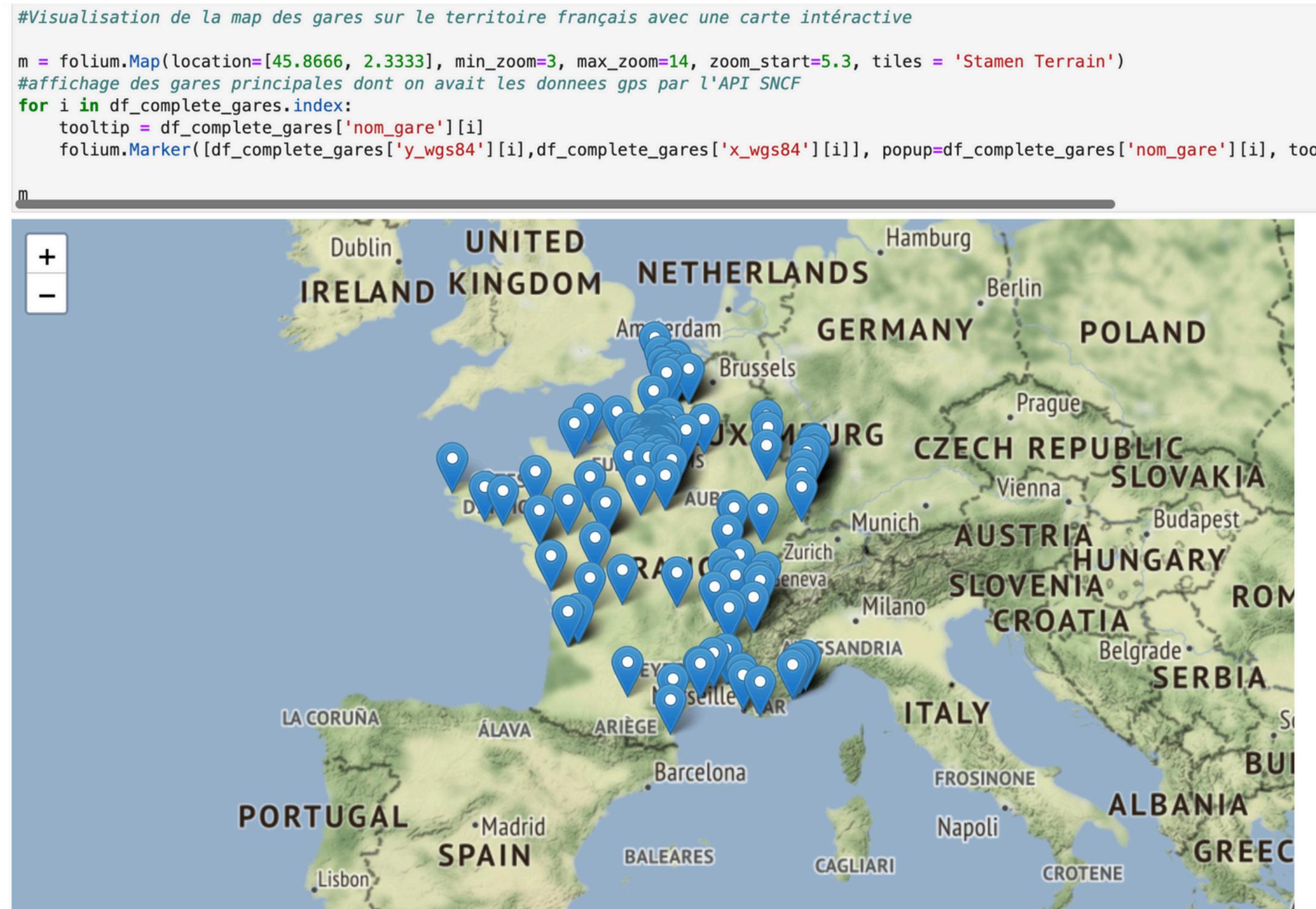
4. Concaténation

```
#Enfin, concaténation des dataframes venant de l'API SNCF et de l'API région IDF
df_complete_gares = pd.concat([df_gares_principales, df_region_idf], ignore_index = True)[['code_uic','nom_gare',
df_complete_gares.head()
```

	code_uic	nom_gare	x_wgs84	y_wgs84	log_total_voyageurs_non_voyageurs_2020
0	87271460	Aéroport Charles de Gaulle 1	2.555964	49.008827	14.593896
1	87741132	Aix-les-Bains le Revard	5.908901	45.688112	14.027522
2	87741132	Aix-les-Bains le Revard	5.908892	45.688219	14.027522
3	87583005	Angoulême	0.164145	45.653643	14.083956
4	87382002	Bécon les Bruyères	2.268813	48.905170	15.299272

Représentation graphique

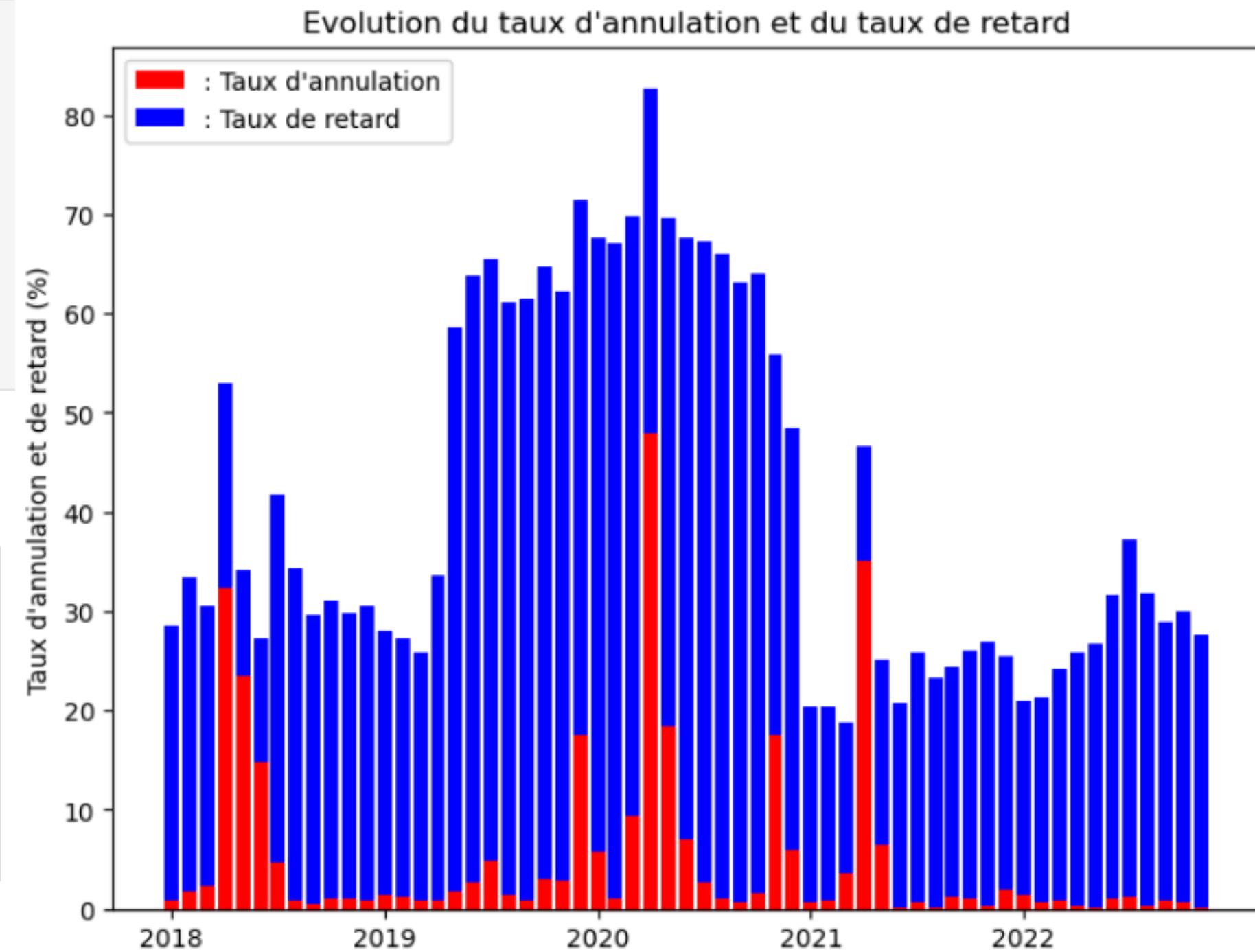
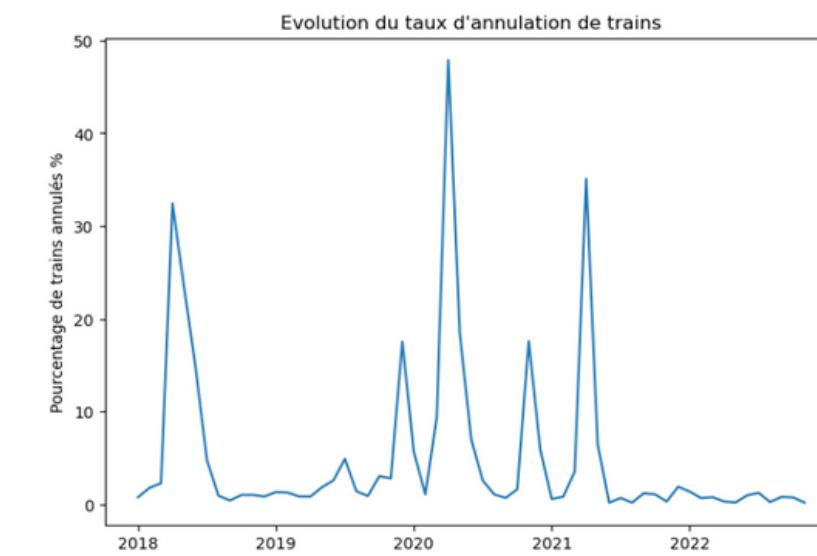
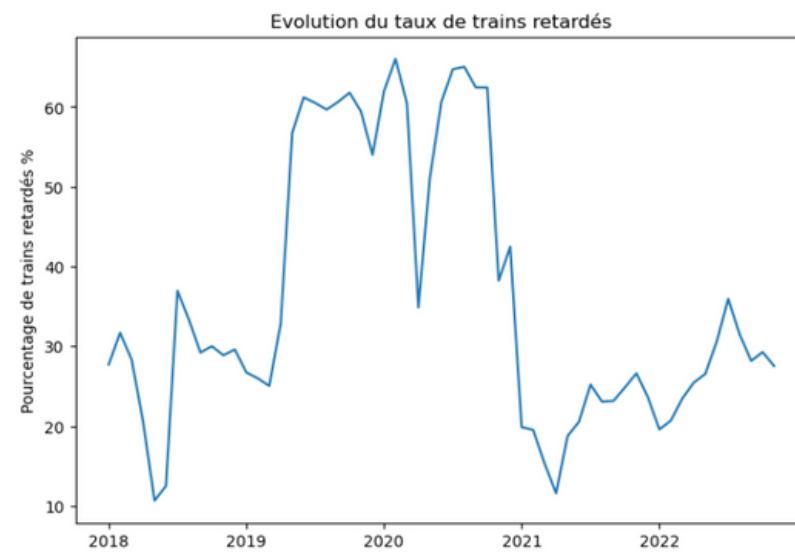
Carte finale (interactive)



Représentation graphique

1. Effet de substitution retard/annulation

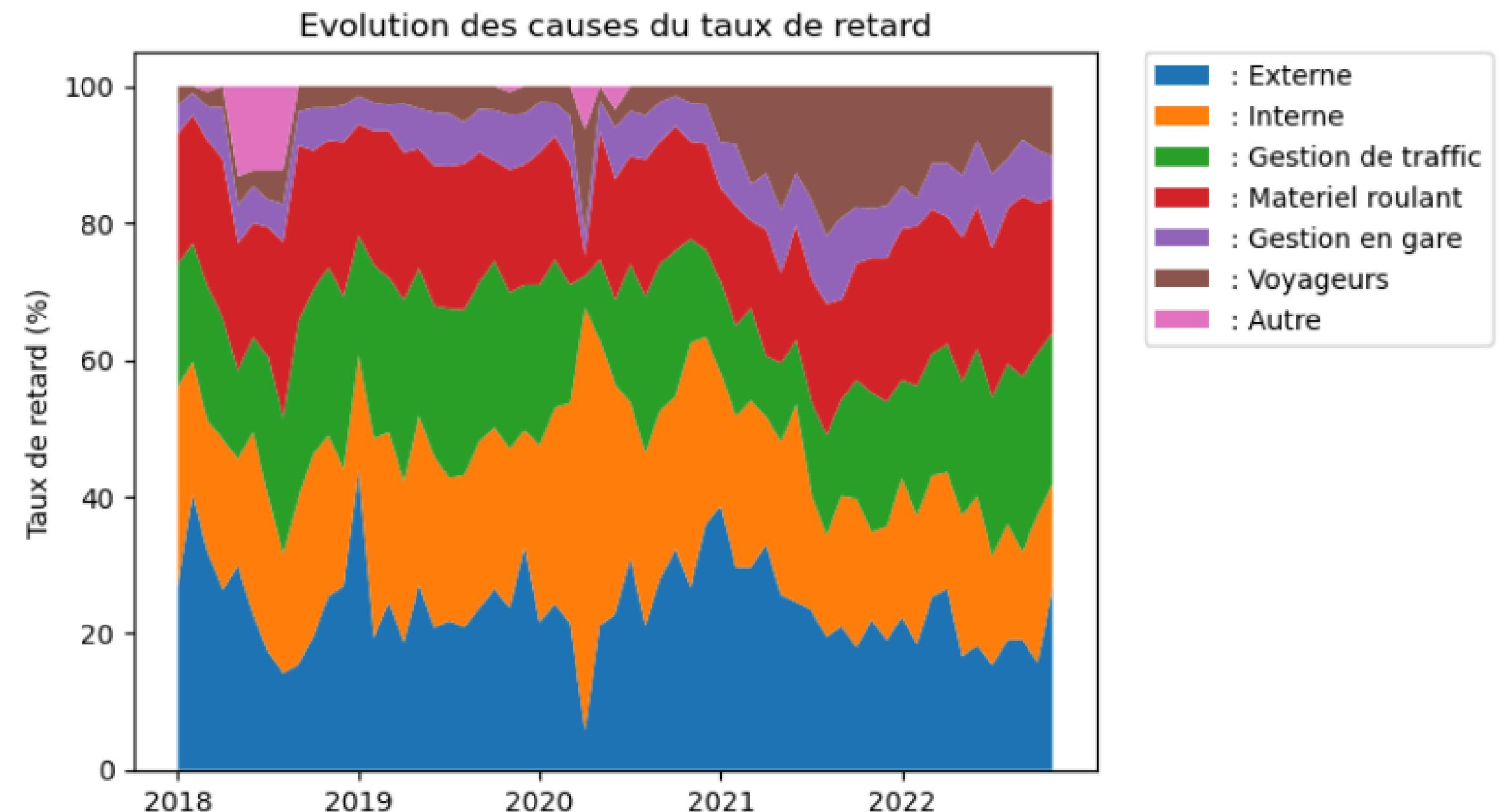
```
ind = np.arange(len(A))
width = 0.8
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
ax.bar(ind, A, width, color='r')
ax.bar(ind, R, width, bottom=A, color='b')
ax.set_ylabel('Taux d\'annulation et de retard (%)')
ax.set_xlabel('Date')
ax.set_title('Evolution du taux d\'annulation et du taux de retard')
ax.legend(labels=[': Taux d\'annulation', ': Taux de retard'])
plt.xticks([0,12,24,36,48],['2018', '2019', '2020', '2021', '2022'])
plt.show()
```



Représentation graphique

2. Evolution des causes de retard

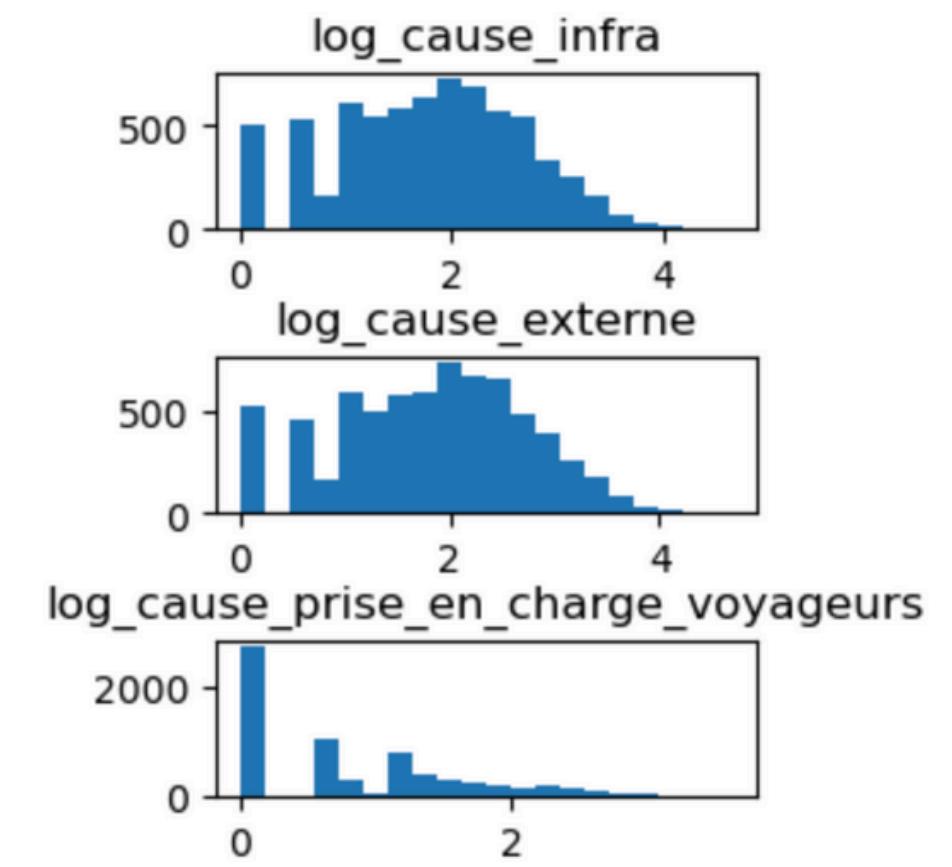
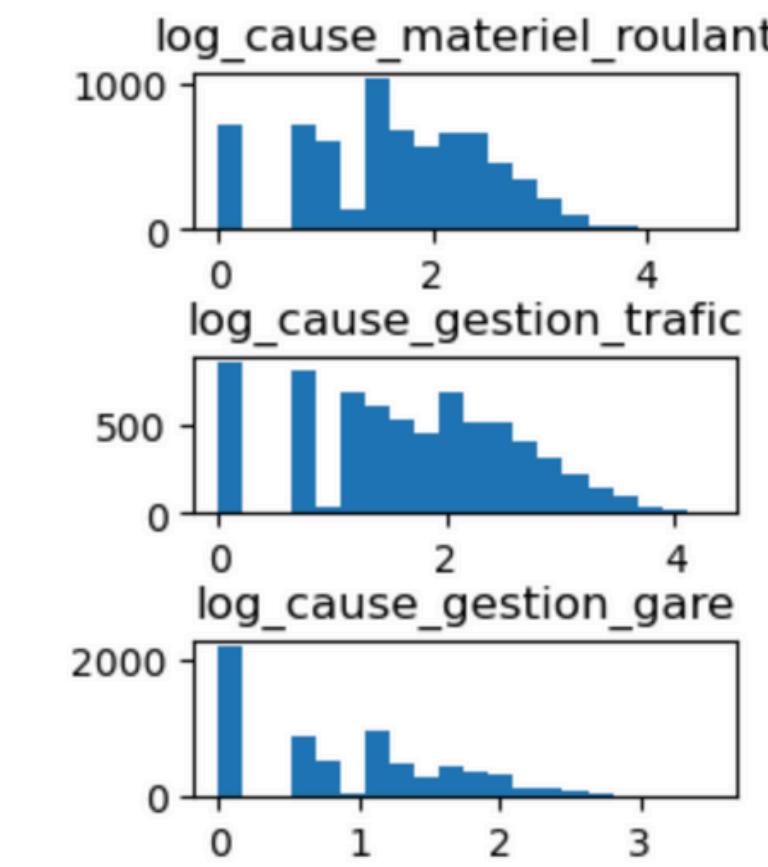
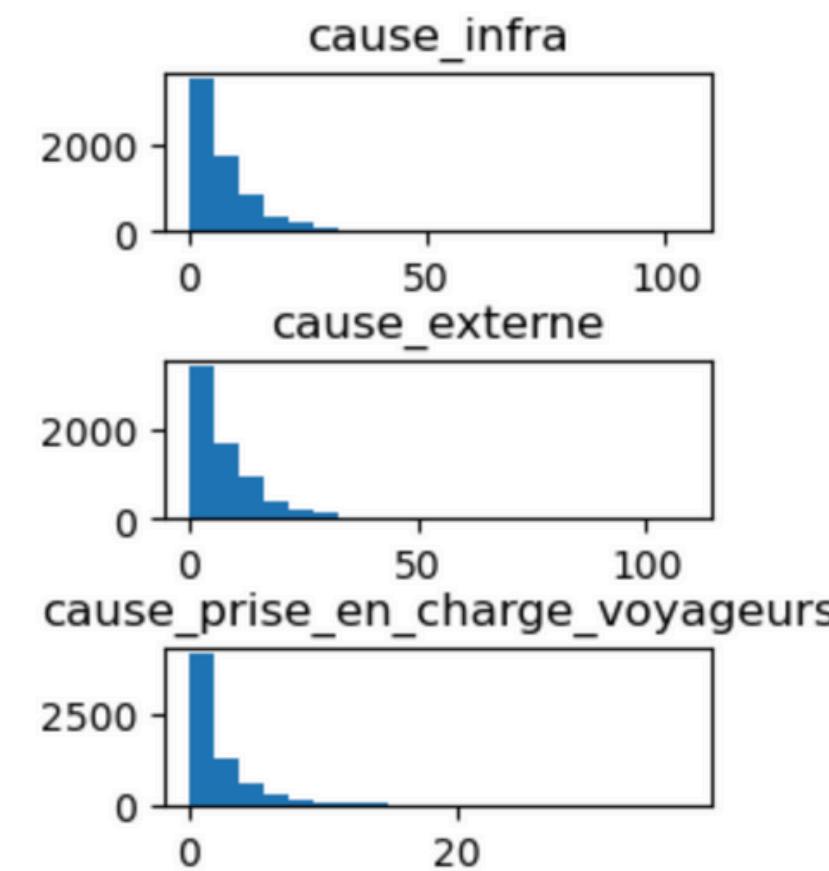
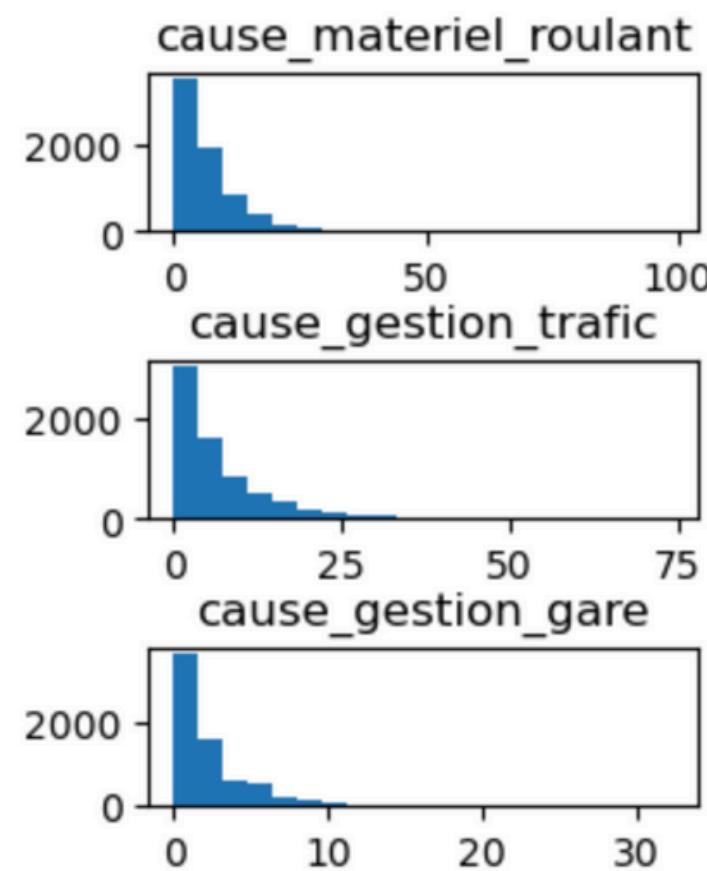
```
#On trace le graphe avec la fonction stackplot
C = [E, I, G, GG, V, O]
inde = range(1,61)
fig, ax = plt.subplots()
ax.stackplot(inde, C)
ax.set_ylabel('Taux de retard (%)')
ax.set_xlabel('Date')
ax.set_title('Evolution des causes du taux de retard')
ax.legend(labels=[': Externe', ': Interne', ': Gestion de traffic', ': Materiel roulant', ': Gestion en gare', ': Voyageurs', ': Autre'], bbox_to_anchor=(1.05, 1), loc='upper left', borderaxespad=0)
plt.xticks([1,13,25,37,49],['2018', '2019', '2020', '2021', '2022'])
plt.show()
```



Modélisation

1. Préparation des données

```
: #Elimination des valeurs excessives  
df_retards = df_retards.drop(df_retards['nb_train_retard_arrivee']>170).index  
df_retards = df_retards.drop(df_retards['nb_train_retard_arrivee']==1).index
```



Modélisation

2. Régression Simple

```
: #Appel de la fonction régression du fichier fonctions.py
```

```
results_s = regression(df_retards[['cause_externe']], df_retards[['nb_train_retard_arrivee']])
print(results_s.summary())
```

OLS Regression Results

```
=====
Dep. Variable:                      y    R-squared:                 0.536
Model:                            OLS   Adj. R-squared:            0.536
Method:                           Least Squares   F-statistic:             8087.
Date:                            Wed, 21 Dec 2022   Prob (F-statistic):        0.00
Time:                             13:23:00      Log-Likelihood:          -30559.
No. Observations:                  6999      AIC:                   6.112e+04
Df Residuals:                     6997      BIC:                   6.114e+04
Df Model:                          1
Covariance Type:                nonrobust
=====
```

	coef	std err	t	P> t	[0.025	0.975]
const	15.8059	0.311	50.894	0.000	15.197	16.415
cause_externe	2.2958	0.026	89.926	0.000	2.246	2.346

```
=====
```

Omnibus:	1992.368	Durbin-Watson:	1.767
Prob(Omnibus):	0.000	Jarque-Bera (JB):	9687.484
Skew:	1.294	Prob(JB):	0.00
Kurtosis:	8.150	Cond. No.	16.6

```
=====
```

Modélisation

3. Régression à plusieurs variables

OLS Regression Results						
Dep. Variable:	y	R-squared:	0.815			
Model:	OLS	Adj. R-squared:	0.815			
Method:	Least Squares	F-statistic:	4412.			
Date:	Wed, 21 Dec 2022	Prob (F-statistic):	0.00			
Time:	13:23:04	Log-Likelihood:	-27334.			
No. Observations:	6999	AIC:	5.468e+04			
Df Residuals:	6991	BIC:	5.474e+04			
Df Model:	7					
Covariance Type:	nonrobust					
		coef	std err	t	P> t	[0.025 0.975]
const		34.7885	0.144	242.005	0.000	34.507 35.070
log_cause_materiel_roulant		3.3471	0.209	16.045	0.000	2.938 3.756
log_cause_infra		6.3395	0.198	32.072	0.000	5.952 6.727
log_cause_gestion_trafic		7.0743	0.203	34.910	0.000	6.677 7.471
log_cause_externe		5.9278	0.194	30.494	0.000	5.547 6.309
log_cause_gestion_gare		2.9437	0.187	15.744	0.000	2.577 3.310
log_cause_prise_en_charge_voyageurs		3.4318	0.173	19.854	0.000	3.093 3.771
nb_train_prevu		4.0072	0.221	18.129	0.000	3.574 4.440
Omnibus:	4237.924	Durbin-Watson:	1.868			
Prob(Omnibus):	0.000	Jarque-Bera (JB):	55758.355			
Skew:	2.691	Prob(JB):	0.00			
Kurtosis:	15.737	Cond. No.	3.65			

Modélisation

4. Prédiction

```
def prediction(x, y, afficher = True):
    #séparation des données en 2 échantillon
    X_train, X_test, y_train, y_test = train_test_split(x,y, test_size=0.2)

    #régression et prédiction
    ols = LinearRegression().fit(X_train, y_train)
    y_pred = ols.predict(X_test)

    #calcul des métriques
    rmse = sklearn.metrics.mean_squared_error(y_test, y_pred, squared = False)
    rsq = sklearn.metrics.r2_score(y_test, y_pred)

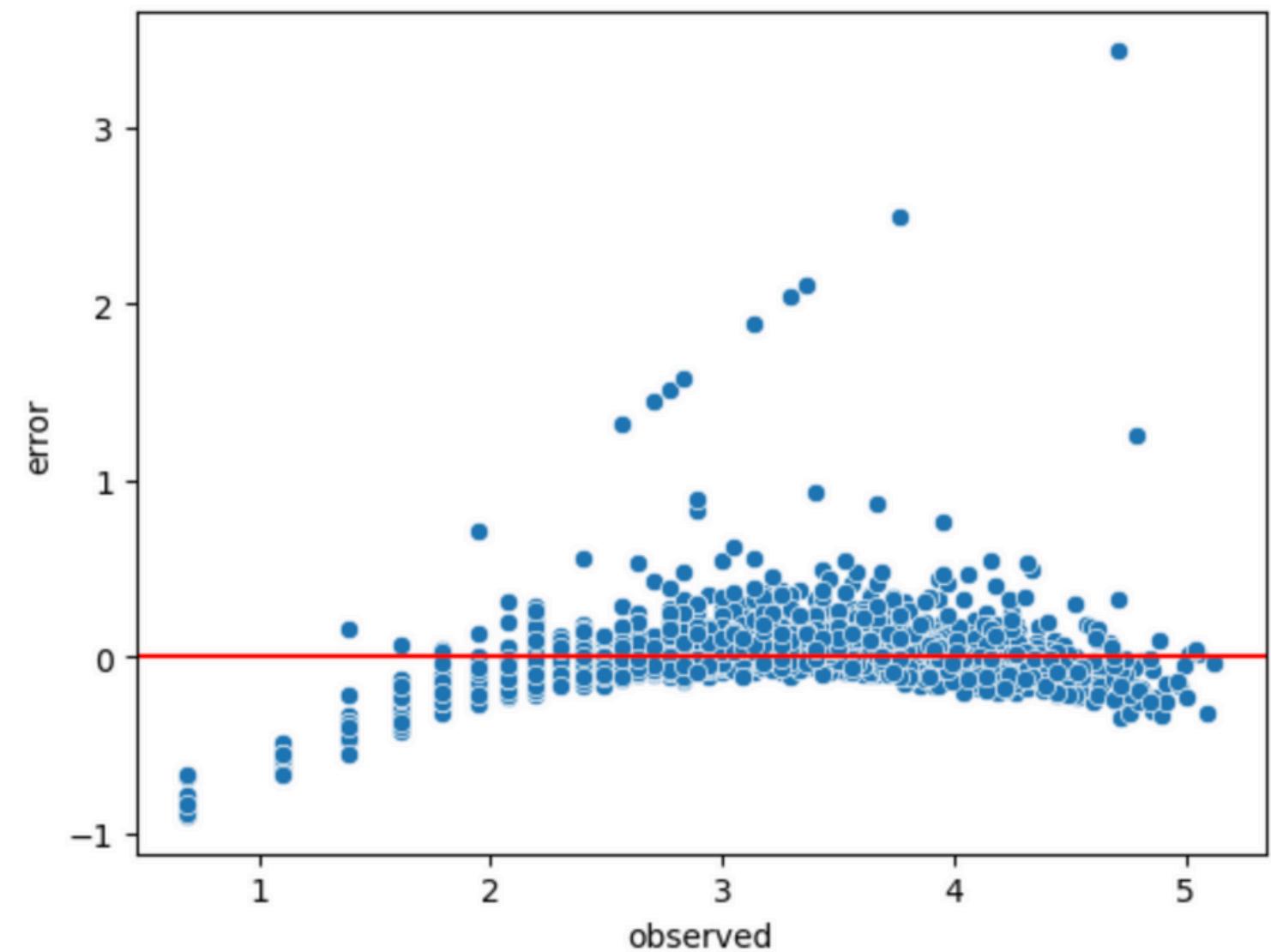
    #nuage de points des valeurs observées
    tempdf = pd.DataFrame({"prediction": y_pred, "observed": y_test,
                           "error": y_test - y_pred})
    if afficher == True :
        print("intercept : ",ols.intercept_)
        print("coeffs : ", ols.coef_)
        print("rsq : ", rsq)
        print("rmse : ", rmse)

        fig = plt.figure()
        g = sns.scatterplot(data = tempdf, x = "observed", y = "error")
        g.axhline(0, color = "red")

    return([X_train, y_train])

x = df_retards[log Causes Retards Large]
y = df_retards['log_nb_train_retard_arrivee']

prediction(x, y)
print()
```



Modélisation

5. Prédiction avec des variables avec le Lasso

	log_cause_materiel_roulant	log_cause_infra	log_cause_gestion_trafic	log_cause_externe
log_cause_materiel_roulant	1.00	0.54	0.59	0.53
log_cause_infra	0.54	1.00	0.57	0.56
log_cause_gestion_trafic	0.59	0.57	1.00	0.55
log_cause_externe	0.53	0.56	0.55	1.00

```
#Estimation du modèle LASSO
X_train = prediction(x, y, afficher = False)[0]
y_train = prediction(x, y, afficher = False)[1]

lasso1 = Lasso(fit_intercept=True, normalize=False, alpha = 0.00655).fit(X_train,y_train)

#selection de variables
features_selec = x.columns[np.abs(lasso1.coef_)>0].tolist()
features_selec

#N'hésitez pas à refaire tourner la cellule si elle ne selectionne aucune ou une seule variable
```

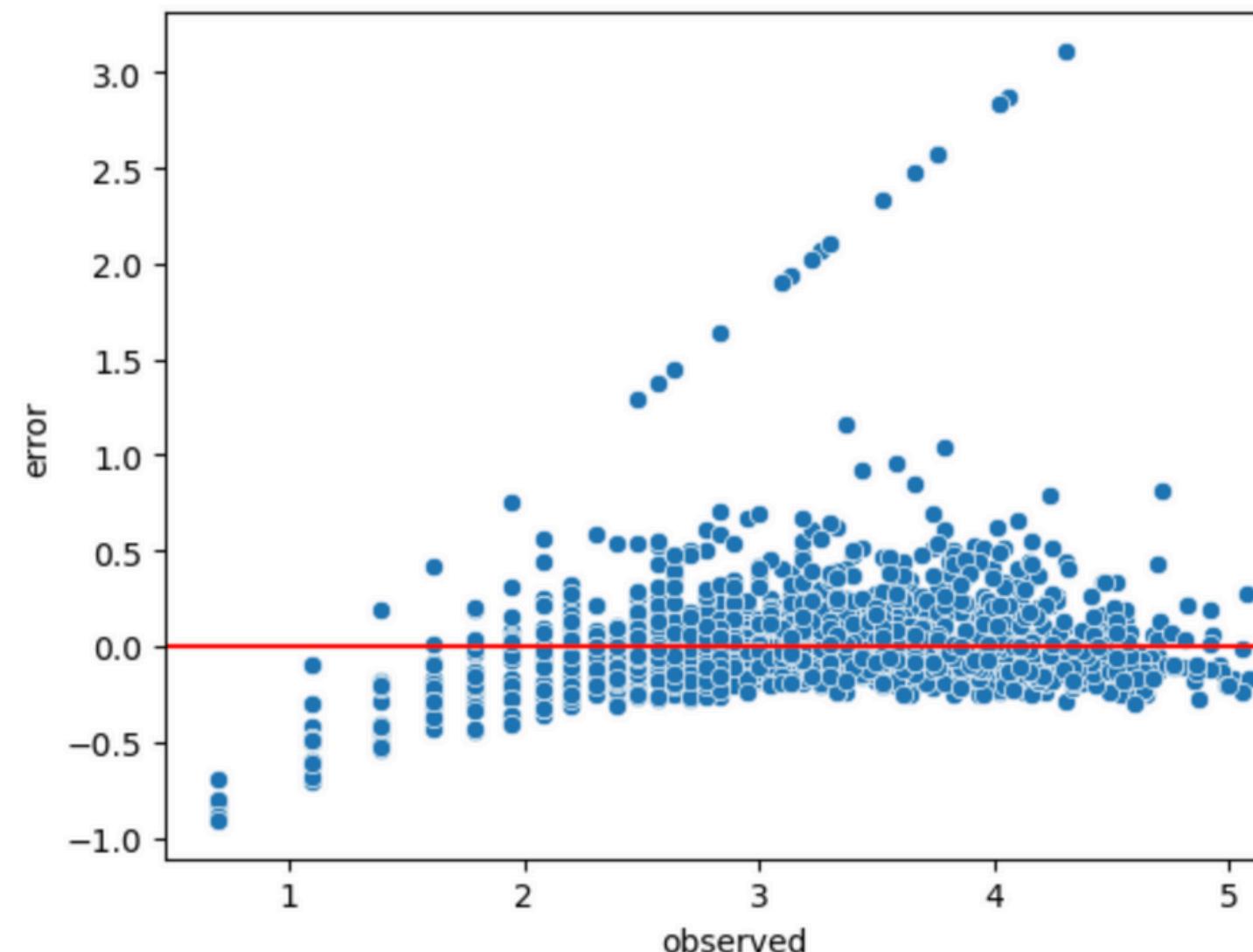
Modélisation

5. Prédiction avec des variables de Lasso

```
x = df_retards[features_selec]
y = df_retards['log_nb_train_retard_arrivee']

prediction(x, y)
print()

intercept : 3.199643228035125
coeffs : [0.25587673 0.25589714 0.26391372 0.27186382]
rsq : 0.8549951137910268
rmse : 0.34263160409932236
```



Mini-jeu avec pygame

1. Classe Player

Inputs	Fonctions
<ul style="list-style-type: none">• position horizontale (avec width)• position verticale (avec height)• --> position donnée par un array (height fixe, width variable) : <code>np.array([width/2, 9*height/10])</code>• largeur• hauteur	<ul style="list-style-type: none">• updatePos pour changer sa position• draw pour afficher le train

Mini-jeu avec pygame

2. Classe Projectile

Inputs	Fonctions
<ul style="list-style-type: none">• position horizontale (avec width)• position verticale (avec height)• vitesse (aléatoire)• type : représente une cause externe, une cause matériel roulant... (sa taille - hauteur, largeur- dépend de son type)• --> position donnée par un array (avec au départ, height = 0 et une position horizontale aléatoire) : np.array([x, 0])	<ul style="list-style-type: none">• isCollision : regarde si un projectile touche le joueur en se basant sur leurs positions (horizontales et verticales)• update : met à jour la position (uniquement verticale) en ajoutant à sa position verticale la vitesse• isOutOfScreen : permet de vérifier si le joueur sort de l'écran de jeu• draw : pour afficher le projectile

Mini-jeu avec pygame

3. Classe Projectiles

Inputs	Fonctions
<ul style="list-style-type: none">• generationRate : proba de générer un projectile à chaque instant• nombreProjectileMax : pour ne pas dépasser un certain nombre de projectiles• width, height : définit la position	<ul style="list-style-type: none">• update : prend la liste des projectiles existant et fait une mise à jour• --> supprime les projectiles sortis de l'écran, génère de nouveaux projectiles de différents types• gameOver : vérifie s'il y a eu collision entre le joueur et les projectiles• drawProjectiles : pour afficher tous les projectiles "vivants"

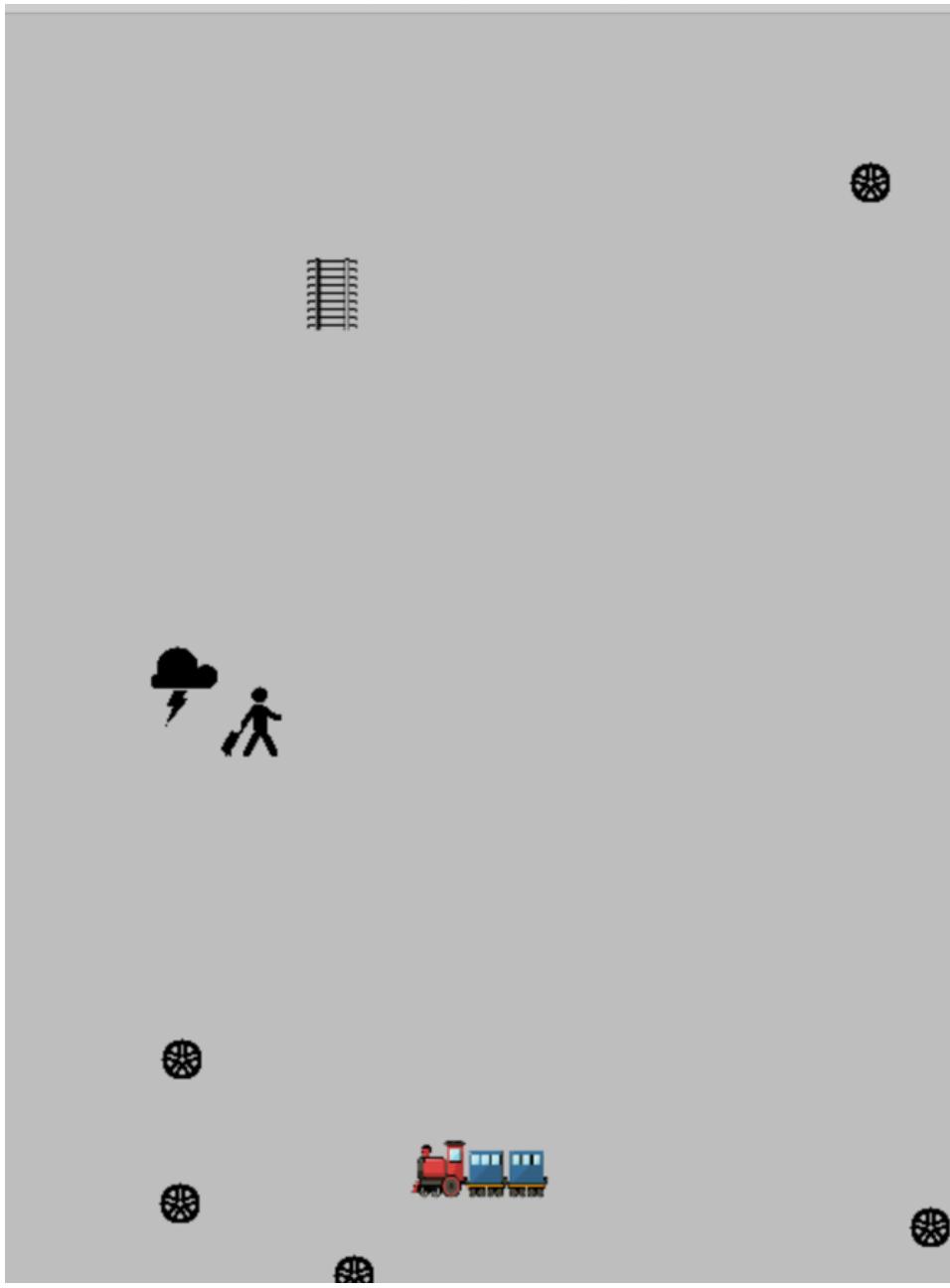
Mini-jeu avec pygame

4. Environnement de jeu : classe world

Inputs	Fonctions
<ul style="list-style-type: none">• generationRate• nombreProjectileMax• width, height : dimensions de l'écran• player, projectiles	<ul style="list-style-type: none">• update_ui : affiche le jeu• play_step_and_draw : tant qu'on ne quitte pas, récupère l'action du joueur (gauche/droite) et met à jour les positions des éléments du jeu. Vérifie s'il y a un évènement Game Over.

Mini-jeu avec pygame

3. Résultat



The screenshot shows a code editor window with a dark theme. The main pane displays the `game.py` script. The terminal pane at the bottom shows the output of the game's execution.

```
game.py > ...
1 #from cgitb import reset
2 import os
3 #os.environ["SDL_VIDEODRIVER"] = "dummy"
4 from os import remove
5 from utils import colors, sp
6 import pygame
7 #from enum import Enum
8 #from collections import namedtuple
9 from player import Player
10 from projectiles import Projectiles, Projectile
11 import time
12 import numpy as np
13 import sys
14
15
16 class world:
17     def __init__(self, generationRate=0.1, nombreProjectilesMax=8, width=480, height=640):
18         self.w = width
19         self.h = height
20         self.player = Player(height, width, 70, 40)
21         self.projectiles = Projectiles(generationRate, nombreProjectilesMax, width, height)
22         self.display = pygame.display.set_mode((self.w, self.h))
23         pygame.display.set_caption('Train')
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
proj bas 510.0
player haut 556.0
player bas 596.0
proj haut 547.0
proj bas 587.0
player haut 556.0
player bas 596.0
collision1
proj haut 540.0
proj bas 560.0
player haut 556.0
player bas 596.0
```