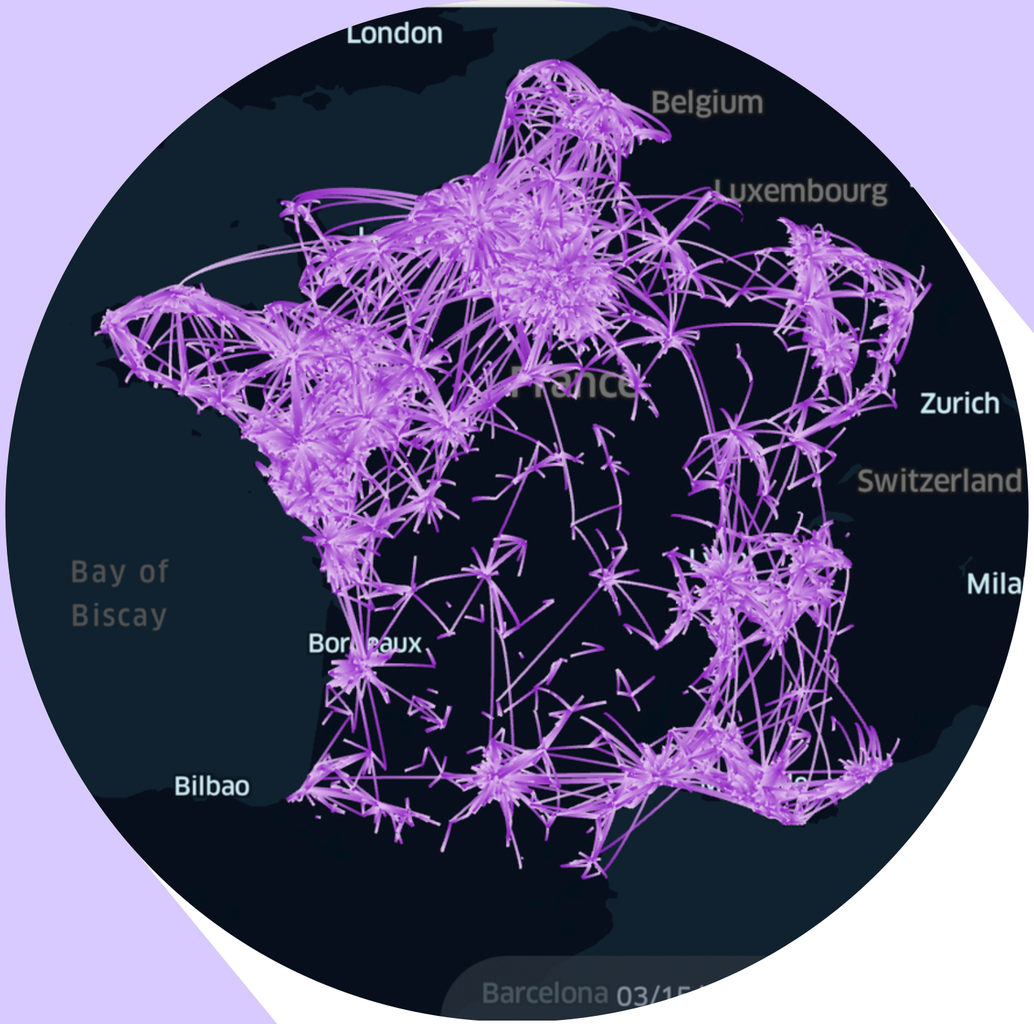
IRONHACK - FINAL PROJECT

CARPOOLING IN FRANCE

Which use ? Which territories ?



IRONHACK DATA ANALYST BOOTCAMP - PARIS MARCH 2023

CLEMENCE LEGRAND

OBJECTIVES

WHAT THE USE ?

Understand the evolution of carpooling, how looks the market today and why people do carpooling.

IN WHICH TERRITORIES ?

See the differences between territories, analyse at a local scale and see where could be the needs.

TYPE STUDY

Descriptive analysis
Territorial dashboard
Predictive model

TABLE OF CONTENTS

Field
journey_id
trip_id
journey_start_datetime
journey_start_date
journey_start_time
journey_start_lat
journey_start_lon
journey_start_insee
journey_start_department
journey_start_towngroup

journey_end_datetime
journey_end_date
journey_end_time
journey_end_lat
journey_end_lon
journey_end_insee
journey_end_department
journey_end_towngroup
number_of_passengers
journey_distance
journey_duration
has_incentive

MY MAIN DATASET

- Carpooling Proof Registry, from DGITM
- Information from the operators
- Carpool journeys in France since 2019 until end of January 2023
- 10 millions of rows, 30 columns

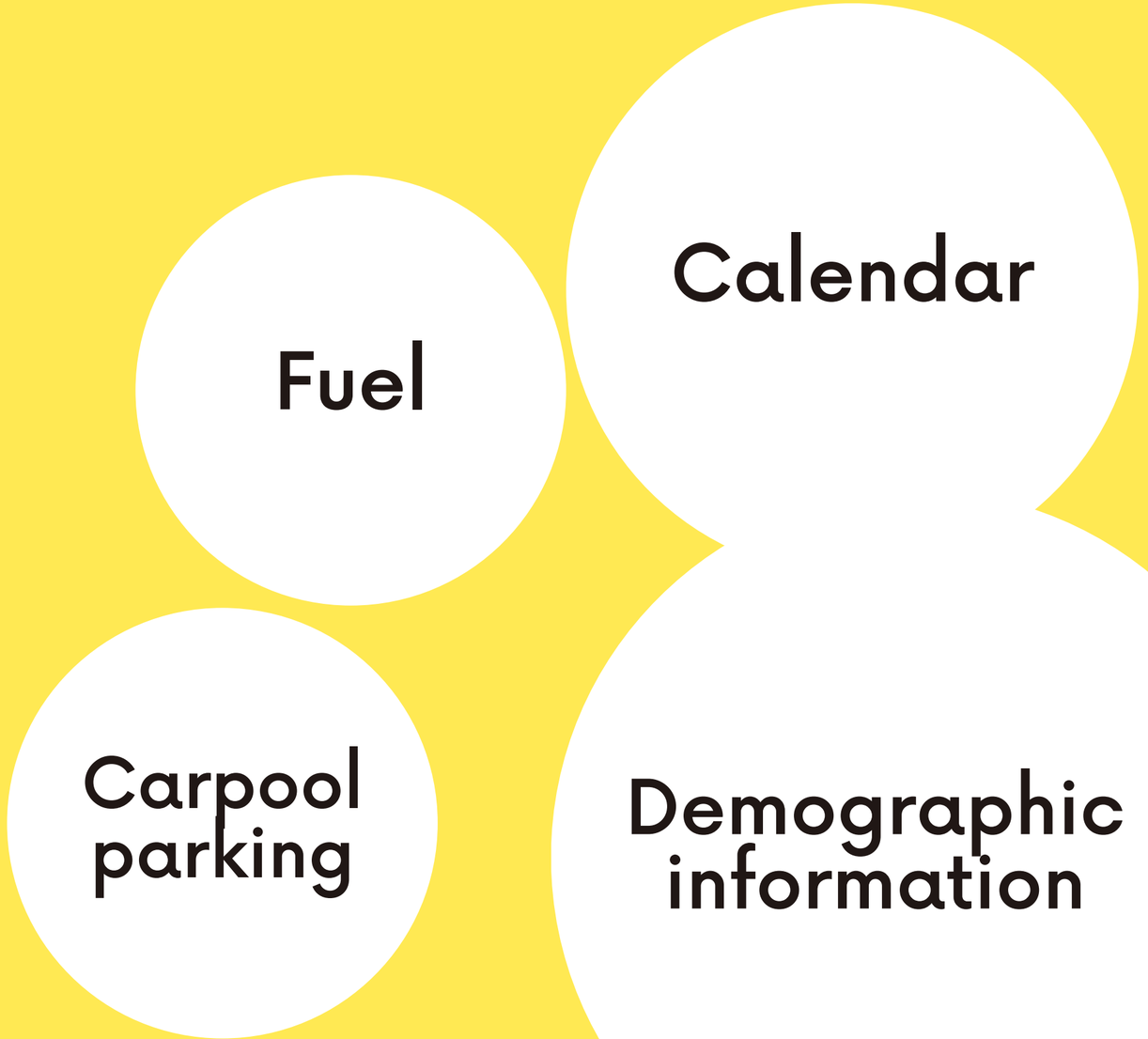
DIMENSIONAL DATASETS

1 Calendar dataset :
- from OpenDataSoft, Ministery of Education
- scholar vacation dates since 2019
- areas of France in vacation
- weekday (added with Python function)

2 Demographic information :
- INSEE
- administrative territories
- population census and density
- percentage of people working in their living city

3 Price of fuels:
- Ministry of Economy
- price per week since jan 2020

4 Carpool areas :
- Ministry of Ecological Transition
- list of parking reserved to carpooling in France



DATA CLEANING

NUMBER OF PASSENGERS

```
trip_id_counts = global_df['trip_id'].value_counts()

# create a new column with the frequencies of trip_id and the number of passenger_seats (we should retire 1 but we w
global_df['nb_passengers'] = global_df['trip_id'].map(trip_id_counts)+global_df["passenger_seats"]#-1

global_df.head()
```

MISSING VALUES

```
#calculate between two times columns
from pandas import datetime
test['journey_start_time'] = pd.to_datetime(test['journey_start_time'])
test['journey_end_time'] = pd.to_datetime(test['journey_end_time'])

test['journey_duration'] = test['journey_end_time'] - test['journey_start_time']

# convert in minutes
test['journey_duration'] = test['journey_duration'].dt.total_seconds() / 60
```

CROSS BORDER TRIPS

```
global_df = global_df[global_df["journey_start_country"]=="France"]
```

```
global_df = global_df.drop("journey_start_country", axis=1)
```

```
global_df = global_df[global_df["journey_end_country"]=="France"]
```

```
global_df = global_df.drop("journey_end_country", axis=1)
```

MAIN DATASET

- From 10,182,966 rows and 30 columns to 9,607,669 rows and 23 columns
- Very good quality
- Calculate the number of passengers
- Missing values
- Inconsistent data
- Cross-border trips

DATA CLEANING

TERRITORIAL DATA

	CODGEO	LIBGEO	DCLT	L_DCLT	NBFLUX_C19_ACTOCC15P
0	1001	L'Abergement-Clémenciat	01001	L'Abergement-Clémenciat	77.008649
1	1001	L'Abergement-Clémenciat	01053	Bourg-en-Bresse	35.518950
2	1001	L'Abergement-Clémenciat	01093	Châtillon-sur-Chalaronne	55.487306
3	1001	L'Abergement-Clémenciat	01159	Feillens	4.954723
4	1001	L'Abergement-Clémenciat	01165	Francheleins	4.820431
5	1001	L'Abergement-Clémenciat	01194	Jassans-Riottier	5.234635
6	1001	L'Abergement-Clémenciat	01263	Montmerle-sur-Saône	4.954723
7	1001	L'Abergement-Clémenciat	01322	Reyrieux	4.954723
8	1001	L'Abergement-Clémenciat	01400	Seillonnaz	4.954723
9	1001	L'Abergement-Clémenciat	01420	Thoissey	4.954723

```
work_commute=work_commute[work_commute["LIBGEO"]==work_commute["L_DCLT"]]
```

```
work_commute.drop(["CODGEO","L_DCLT"],axis=1,inplace=True)
```

```
work_commute.rename(columns={"LIBGEO":"city_name","DCLT":"id_city","NBFLUX_C19_ACTOCC15P": "no_commute_pop"},inplac
```

```
work_commute
```

	city_name	id_city	no_commute_pop
0	L'Abergement-Clémenciat	01001	77.008649
33	L'Abergement-de-Varey	01002	10.402373
50	Ambérieu-en-Bugey	01004	2420.133858
243	Ambérieux-en-Dombes	01005	152.303591
320	Ambronay	01007	180.000000

DATA CLEANING

FUEL PRICES

	Date	Gazole	Super SP95	Super SP95-E10	Super SP98	Superéthanol E85	GPLc
0	Date	Gazole	Super SP95	Super SP95-E10	Super SP98	Superéthanol E85	GPLc
1	2020-01-03 00:00:00	1.4841	1.5365	1.5186	1.6006	0.6744	0.858
2	2020-01-10 00:00:00	1.4908	1.5483	1.5276	1.6097	0.6746	0.8531
3	2020-01-17 00:00:00	1.4791	1.5401	1.5181	1.6019	0.6752	0.8494
4	2020-01-24 00:00:00	1.4635	1.5343	1.5114	1.5938	0.6767	0.8515

```
gaz_df["mean_price"]=gaz_df.mean(axis = 1)
```

	Date	Gazole	Super SP95	Super SP95-E10	Super SP98	mean_price
0	2020-01-03	1.4841	1.5365	1.5186	1.6006	1.53495
1	2020-01-10	1.4908	1.5483	1.5276	1.6097	1.54410
2	2020-01-17	1.4791	1.5401	1.5181	1.6019	1.53480
3	2020-01-24	1.4635	1.5343	1.5114	1.5938	1.52575
4	2020-01-31	1.4343	1.516	1.4914	1.5775	1.50480

CARPOOL AREAS

```
park_df["id_lieu"].nunique()
```

```
8799
```

```
park_df.isnull().sum()
```

```
id_lieu        0
id_local    8630
nom_lieu       0
ad_lieu     1407
com_lieu       3
insee          0
type           0
date_maj       0
ouvert         0
source       173
Xlong          0
Ylat           0
nbre_pl     6821
nbre_pmr    7876
duree       8762
horaires    8606
proprio     7963
lumiere     2841
comm        8507
dtype: int64
```

```
counts=pd.Series(park_df.isnull().sum())
```

```
columns_to_drop=counts[counts>200].index.tolist()
```

```
columns_to_drop
```

```
['id_local',
 'ad_lieu',
 'nbre_pl',
 'nbre_pmr',
 'duree',
 'horaires',
 'proprio',
 'lumiere',
 'comm']
```

```
column_indices = [park_df.columns.get_loc(col) for col in columns_to_drop]

park_df = park_df.drop(park_df.columns[column_indices], axis=1)
```

SCHEDULE

```
schedule_df["Weekday"]=schedule_df['Date'].dt.day_name()
```

```
schedule_df.head()
```

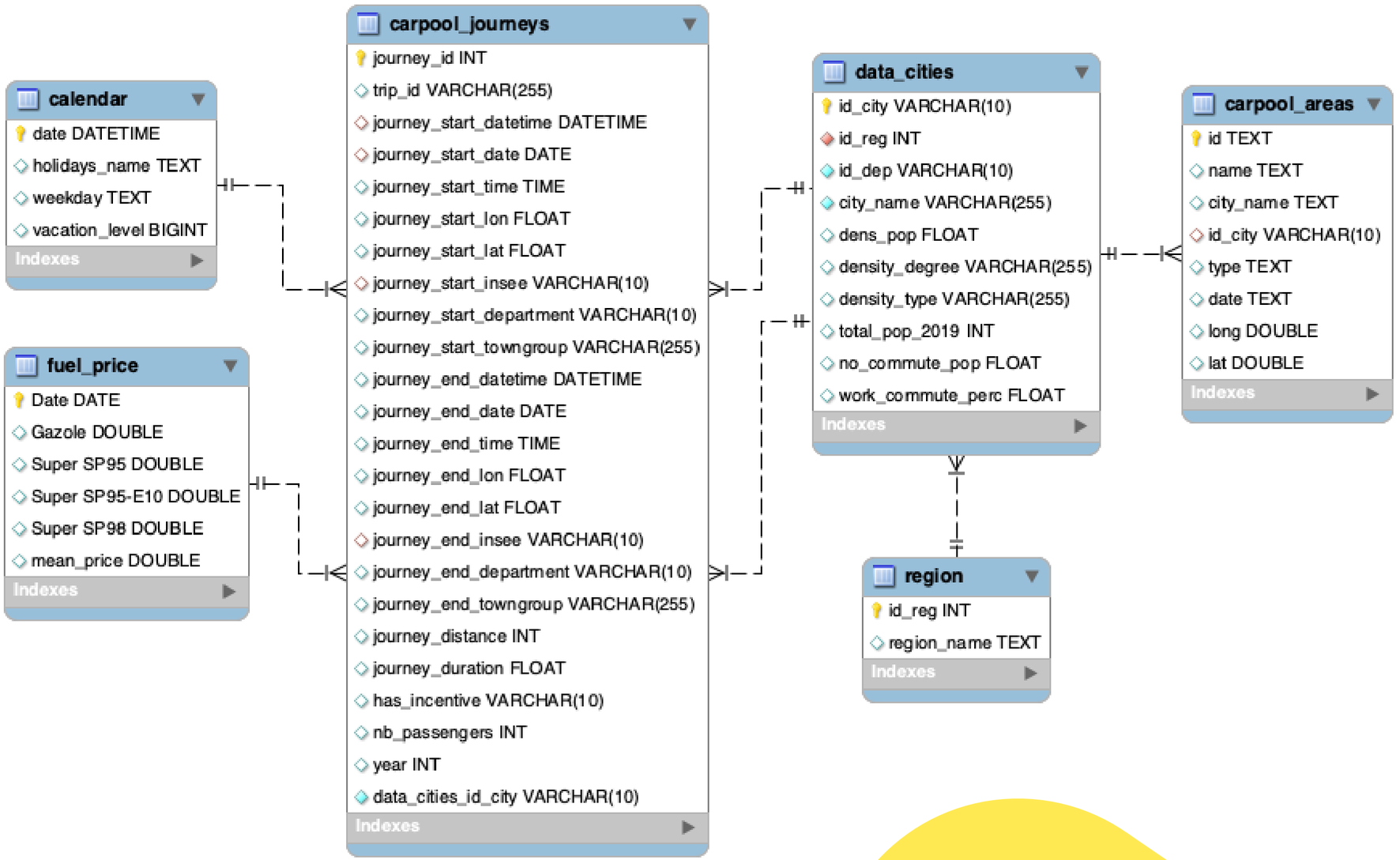
	Date	Calendrier Zone A	Calendrier Zone B	Calendrier Zone C	Nom de la période	timestamp_unix	Weekday
0	2019-01-02	Vacances de Noël	Vacances de Noël	Vacances de Noël	Vacances de Noël	1546387200	Wednesday
1	2019-01-27	Hors Vacances	Hors Vacances	Hors Vacances	NaN	1548547200	Sunday
2	2019-02-26	Vacances d'hiver	Hors Vacances	Vacances d'hiver	Vacances d'hiver	1551139200	Tuesday
3	2019-03-08	Hors Vacances	Hors Vacances	Vacances d'hiver	Vacances d'hiver	1552003200	Friday
4	2019-04-02	Hors Vacances	Hors Vacances	Hors Vacances	NaN	1554163200	Tuesday

```
schedule_df["vacation_level"]=schedule_df["Calendrier Zone A"]+schedule_df["Calendrier Zone B"]+sc
```

```
schedule_df.head(20)
```

	id_date	Calendrier Zone A	Calendrier Zone B	Calendrier Zone C	Nom de la période	Weekday	vacation_level
159	2019-01-01	1	1	1	Vacances de Noël	Tuesday	3
0	2019-01-02	1	1	1	Vacances de Noël	Wednesday	3
146	2019-01-03	1	1	1	Vacances de Noël	Thursday	3
338	2019-01-04	1	1	1	Vacances de Noël	Friday	3
169	2019-01-05	1	1	1	Vacances de Noël	Saturday	3
222	2019-01-06	1	1	1	Vacances de Noël	Sunday	3
147	2019-01-07	0	0	0	NaN	Monday	0
308	2019-01-08	0	0	0	NaN	Tuesday	0
205	2019-01-09	0	0	0	NaN	Wednesday	0

	id_date	Nom de la période	Weekday	vacation_level
159	2019-01-01	Vacances de Noël	Tuesday	3
0	2019-01-02	Vacances de Noël	Wednesday	3
146	2019-01-03	Vacances de Noël	Thursday	3
338	2019-01-04	Vacances de Noël	Friday	3
169	2019-01-05	Vacances de Noël	Saturday	3



ENTITY RELATIONAL DIAGRAM

TERRITORIAL INFORMATION

```sql
INSERT INTO data_cities (id_city, id_reg, id_dep, city_name, dens_pop, density_degree, density_type, total_pop_2019, no_commute_pop)
SELECT
    ct.id_city,
    ct.id_reg,
    ct.id_dep,
    ct.city_name,
    d.dens_pop,
    dl.density_degree,
    dl.density_type,
    dl.total_pop_2019,
    wc.no_commute_pop
FROM code_territories ct
JOIN density d ON ct.id_city = d.id_city
JOIN density_labels dl ON ct.id_city = dl.id_city
LEFT JOIN work_commute wc ON ct.id_city = wc.id_city;

## I want to create an indicator in data_cities to have the proportion of people movinf to another city to work
ALTER TABLE data_cities
ADD COLUMN work_commute_perc FLOAT;

UPDATE data_cities
SET work_commute_perc=(total_pop_2019-no_commute_pop)/total_pop_2019*100;
```

MAIN DATASET

```sql
CREATE TABLE IF NOT EXISTS carpool_journeys (
    journey_id int not null unique,
    trip_id varchar(255),
    journey_start_datetime datetime,
    journey_start_date date,
    journey_start_time time,
    journey_start_lon float,
    journey_start_lat float,
    journey_start_insee varchar(10),
    journey_start_department varchar(10),
    journey_start_towngroup varchar (255),
    journey_end_datetime datetime,
    journey_end_date date,
    journey_end_time time,
    journey_end_lon float,
    journey_end_lat float,
    journey_end_insee varchar(10),
    journey_end_department varchar(10),
    journey_end_towngroup varchar(255),
    journey_distance int,
    journey_duration float,
    has_incentive varchar(10),
    nb_passengers int,
    year int,
    PRIMARY KEY (journey_id)
    );
```

```sql
INSERT INTO carpool_journeys (
    journey_id,
    trip_id,
    journey_start_datetime,
    journey_start_date,
    journey_start_time,
    journey_start_lon,
    journey_start_lat,
    journey_start_insee,
    journey_start_department,
    journey_start_towngroup,
    journey_end_datetime,
    journey_end_date,
    journey_end_time,
    journey_end_lon,
    journey_end_lat,
    journey_end_insee,
    journey_end_department,
    journey_end_towngroup,
    journey_distance,
    journey_duration,
    has_incentive,
    nb_passengers,
    year)
SELECT * FROM carpool_2019;
```

SQL

- Import with SQLAlchemy
- Creation of the tables

```sql
#Carpooling journeys per year and per incentives
SELECT year, count(*) AS total_journeys, count(CASE WHEN has_incentive="OUI" THEN 1 END) AS with_incentives
FROM carpool_journeys
GROUP BY year
ORDER BY year;
```

year	total_journeys	with_incentives
2019	751622	570696
2020	1712378	1083626
2021	1477834	1398116
2022	4911173	4681458
2023	754662	733718

```sql
#I want to see the territorial group town where there is more carpool journeys in absolute values
SELECT cj.journey_start_towngroup, count(journey_id) as total_journeys
FROM carpool_journeys cj
WHERE cj.journey_start_towngroup is not null
GROUP BY cj.journey_start_towngroup
ORDER BY total_journeys DESC
LIMIT 15;
```

journey_start_towngroup	total_journeys
Ile-De-France Mobilites	3303739
Métropole du Grand Paris	599461
Metropole Rouen Normandie	489663
Montpellier Mediterranee Metropole	239269
Nantes Metropole	172359
Syndicat Mixte Des Transports En Commun De...	125243
CA Communauté Paris-Saclay	124964
Metz Metropole	107374
CA Grand Paris Sud Seine Essonne Sénart	104695
CA du Beauvaisis	96691
CU Angers Loire Métropole	93143
Metropole Europeenne De Lille	80687
Metropole Aix-Marseille-Provence	79628
CA Cœur d'Essonne Agglomération	78786
Angers Loire Metropole	74970

```sql
#Look at the number of carpool journeys per capita depending of the type of density territories
SELECT dc.density_type, dc.density_degree,
count(cj.journey_id)/sum(total_pop_2019)*1000 as total_journeys_1000inhab
FROM carpool_journeys cj
JOIN data_cities dc  ON dc.id_city=cj.journey_start_insee
GROUP BY dc.density_type, dc.density_degree
ORDER BY total_journeys_1000inhab DESC;
```

density_type	density_degree	total_journeys_1000inhab
Rural à habitat très dispersé	7	1.5413
Rural à habitat dispersé	6	0.5978
Bourgs ruraux	5	0.2578
Ceintures urbaines	4	0.1647
Petites villes	3	0.0959
Centres urbains intermédiaires	2	0.0346
Grands centres urbains	1	0.0063

SQL QUERIES

- Carpooling is increasing
- The size of metropoles does not determine the number of journeys.
- Per inhabitant, there are more journeys in rural areas.

```sql
#Top 15 of the departments with most carpool journeys per capita
SELECT pd.id_dep as department,
    pd.total_pop_dep,
    count(cj.journey_id),
    count(cj.journey_id)/pd.total_pop_dep*1000 as journeys_per_1000inhab
FROM population_department pd
INNER JOIN carpool_journeys cj ON cj.journey_start_department=pd.id_dep
-- WHERE cj.journey_id<936378
GROUP BY department, total_pop_dep
ORDER BY journeys_per_1000inhab DESC
LIMIT 15;
```

departme...	total_pop_d...	count(cj.journey_...	journeys_per_1000inh...
91	1301659	974008	748.2820
78	1448207	707442	488.4951
76	1255633	560902	446.7086
77	1421197	611286	430.1205
94	1407124	571798	406.3594
92	1624357	609775	375.3947
95	1249674	441881	353.5970
49	818273	278038	339.7864
53	307062	93563	304.7039
34	1175623	326885	278.0526
93	1644903	413701	251.5048
60	829419	195136	235.2683
44	1429272	306737	214.6107
85	685442	129608	189.0867
75	2165423	401764	185.5360

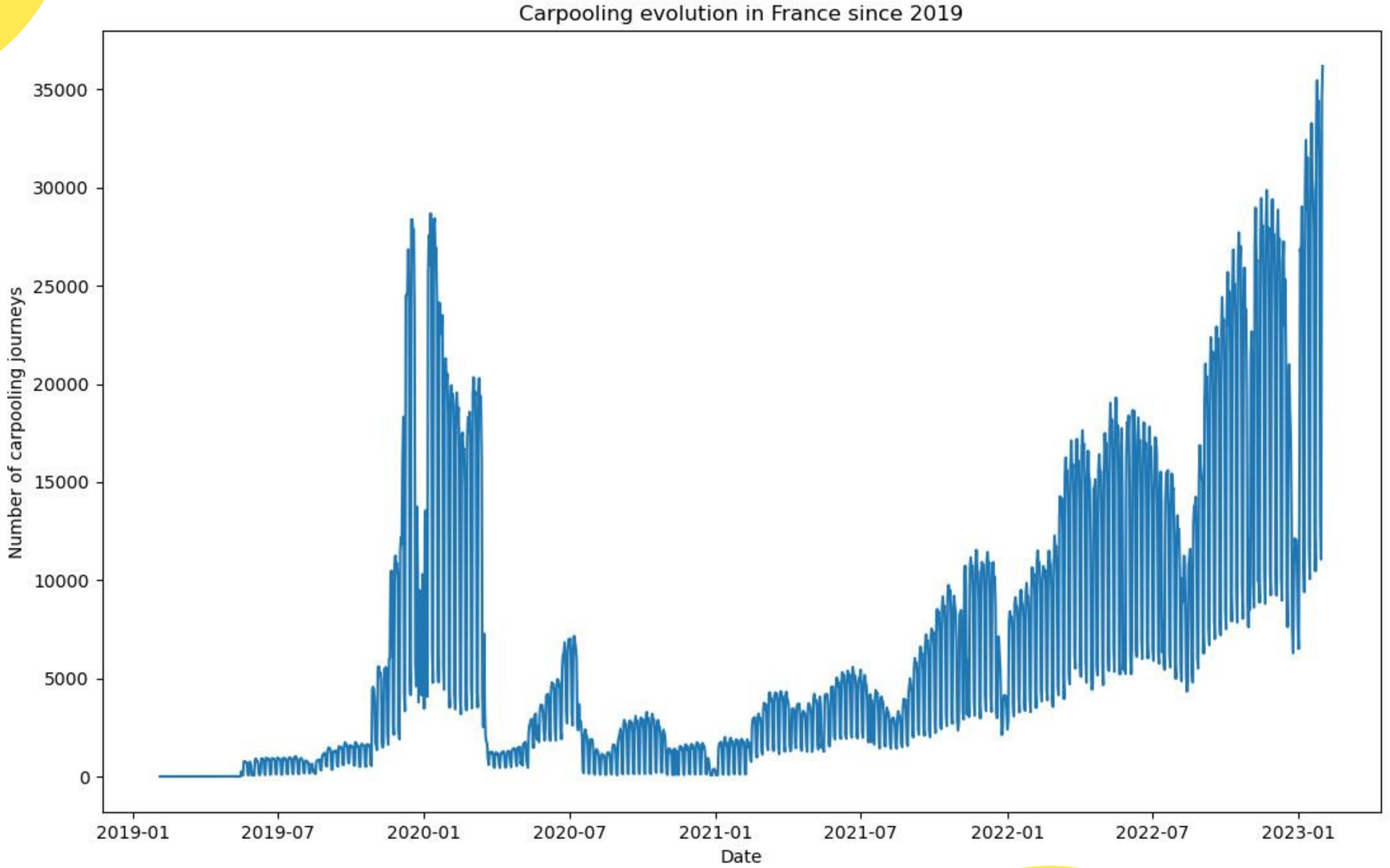
```sql
#Top 15 of the departments with high work-commute population
SELECT id_dep as department,
sum(no_commute_pop) as no_commute_pop,
sum(total_pop_2019) as total_pop_dep,
(sum(total_pop_2019)-sum(no_commute_pop))/sum(total_pop_2019)*1000 as commute_people_per_1000_inhab
FROM data_cities
GROUP BY department
ORDER BY commute_people_per_1000_inhab DESC
LIMIT 15;
```

department	no_commute_pop	total_pop_dep	commute_people_per_1000_habitant
69	127856.70292425156	1875747	931.8369146136171
95	97319.38430213928	1249674	922.1241825450963
91	108999.2395772934	1301659	916.261294565402
77	119161.27219343185	1421197	916.154289522542
93	140418.6276550293	1644903	914.6340983905864
78	127067.91999721527	1448207	912.258454767022
94	123477.57485961914	1407124	912.2482632236965
13	181778.7587928772	2043110	911.0284033689438
62	131353.9445066452	1465278	910.3556154486417
60	76781.72447490692	829419	907.4270971910374
54	70550.85018777847	733760	903.8502368788453
57	102233.73611211777	1046543	902.3129139346231
27	59754.1141834259	599507	900.3279124623634
59	263542.6309623718	2608346	898.9617823086462
92	166278.53475952148	1624357	897.6342424974796

SQL QUERIES

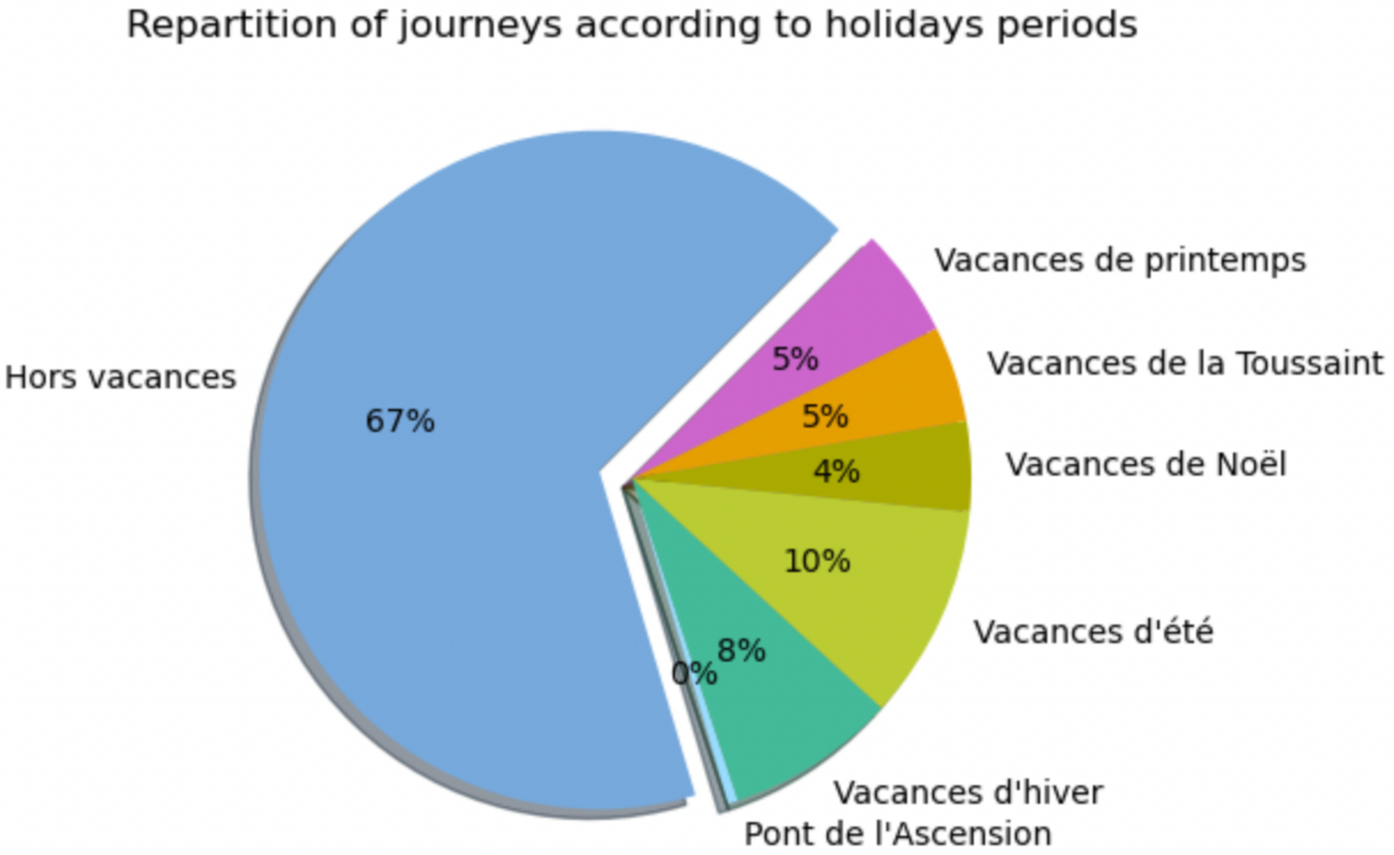
- Our database can show where there are a lot commuting people but low number of carpool journeys

GLOBAL INSIGHTS ABOUT CARPOOLING IN FRANCE SINCE 2019



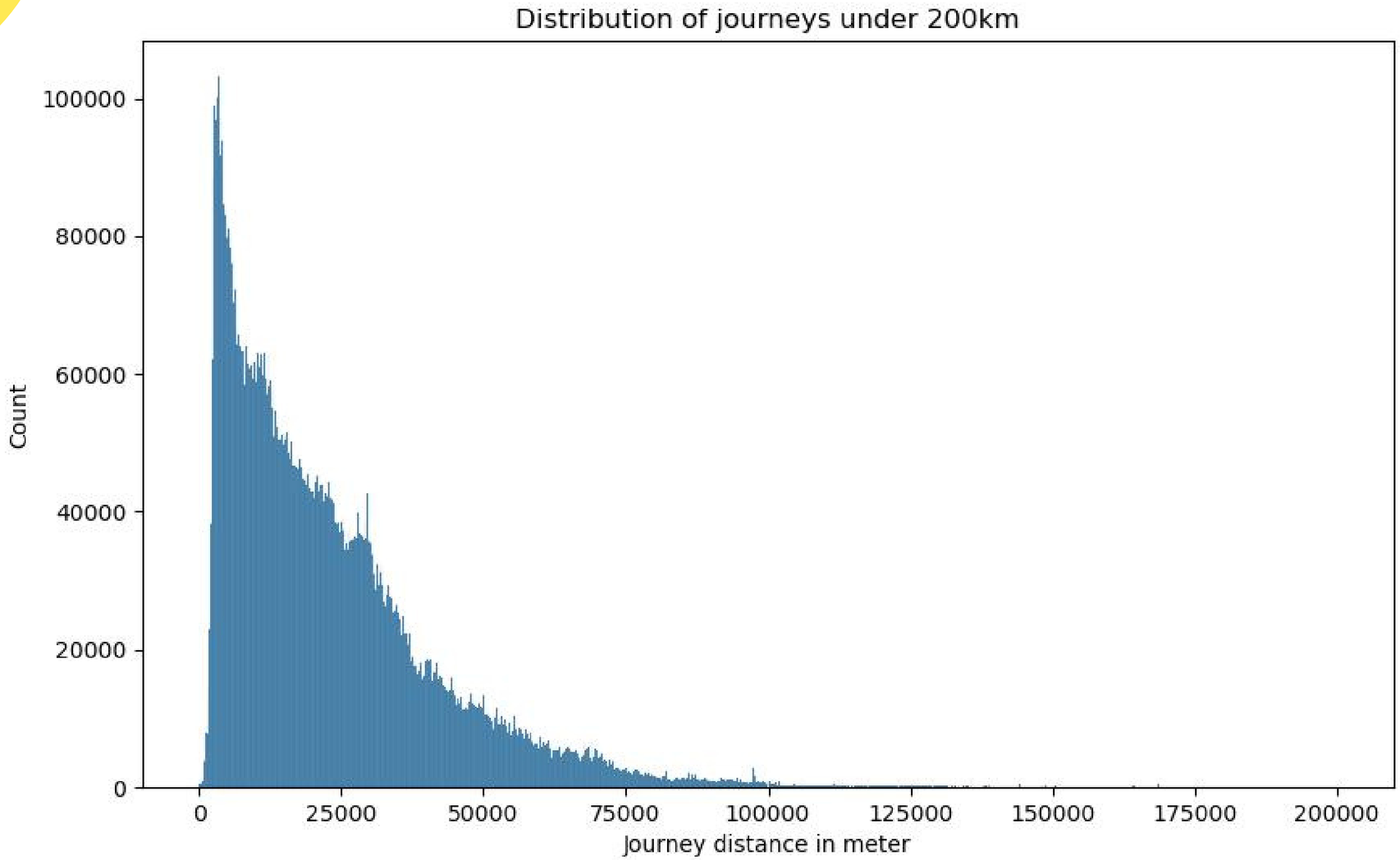
GLOBAL EVOLUTION

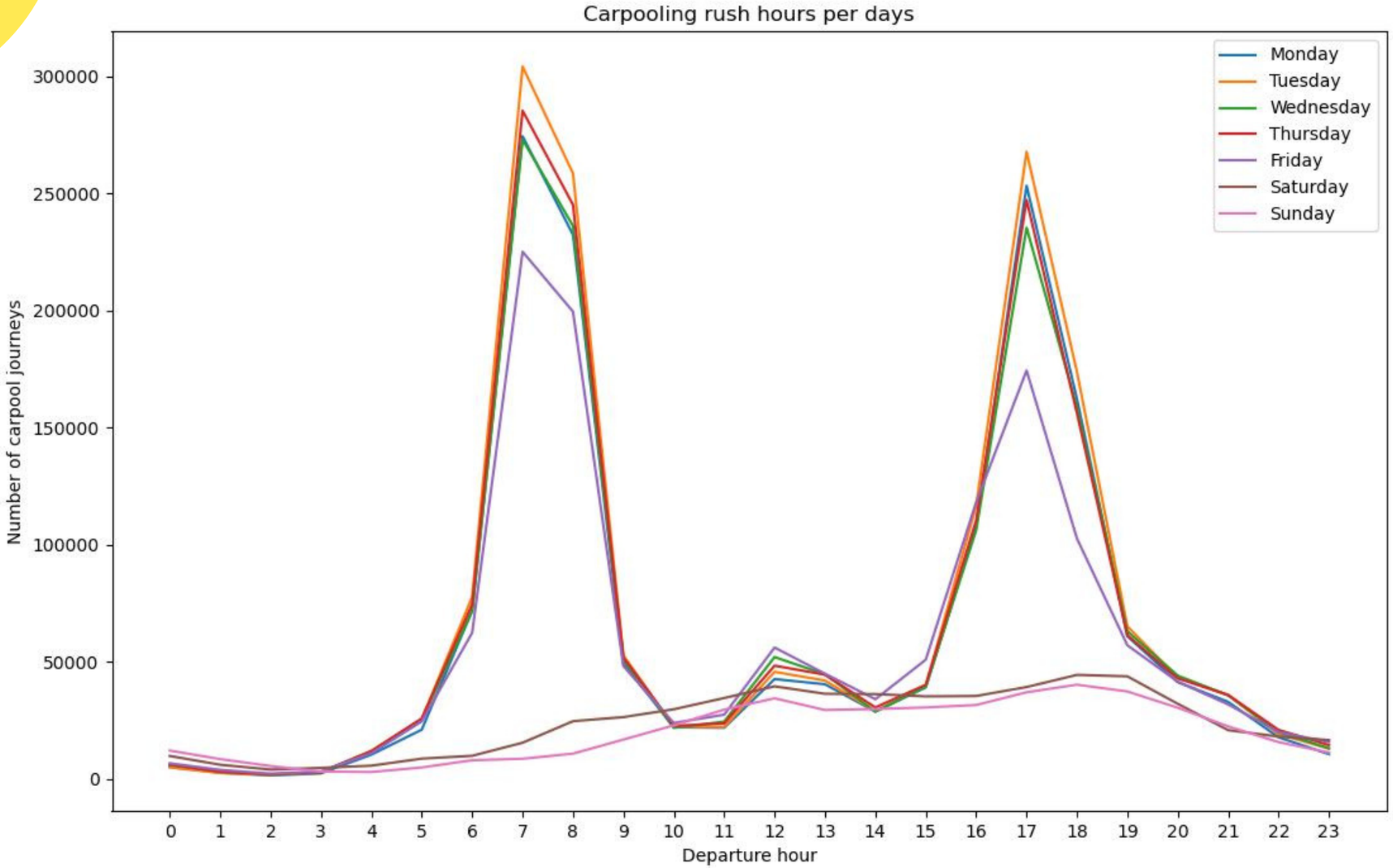
CALENDAR


Repartition of journeys according to holidays periods
Hors vacances
67%
Vacances de printemps
5%
Vacances de la Toussaint
5%
Vacances de Noël
4%
Vacances d'été
10%
Vacances d'hiver
8%
Pont de l'Ascension
0%

SHORT-DISTANCE JOURNEYS

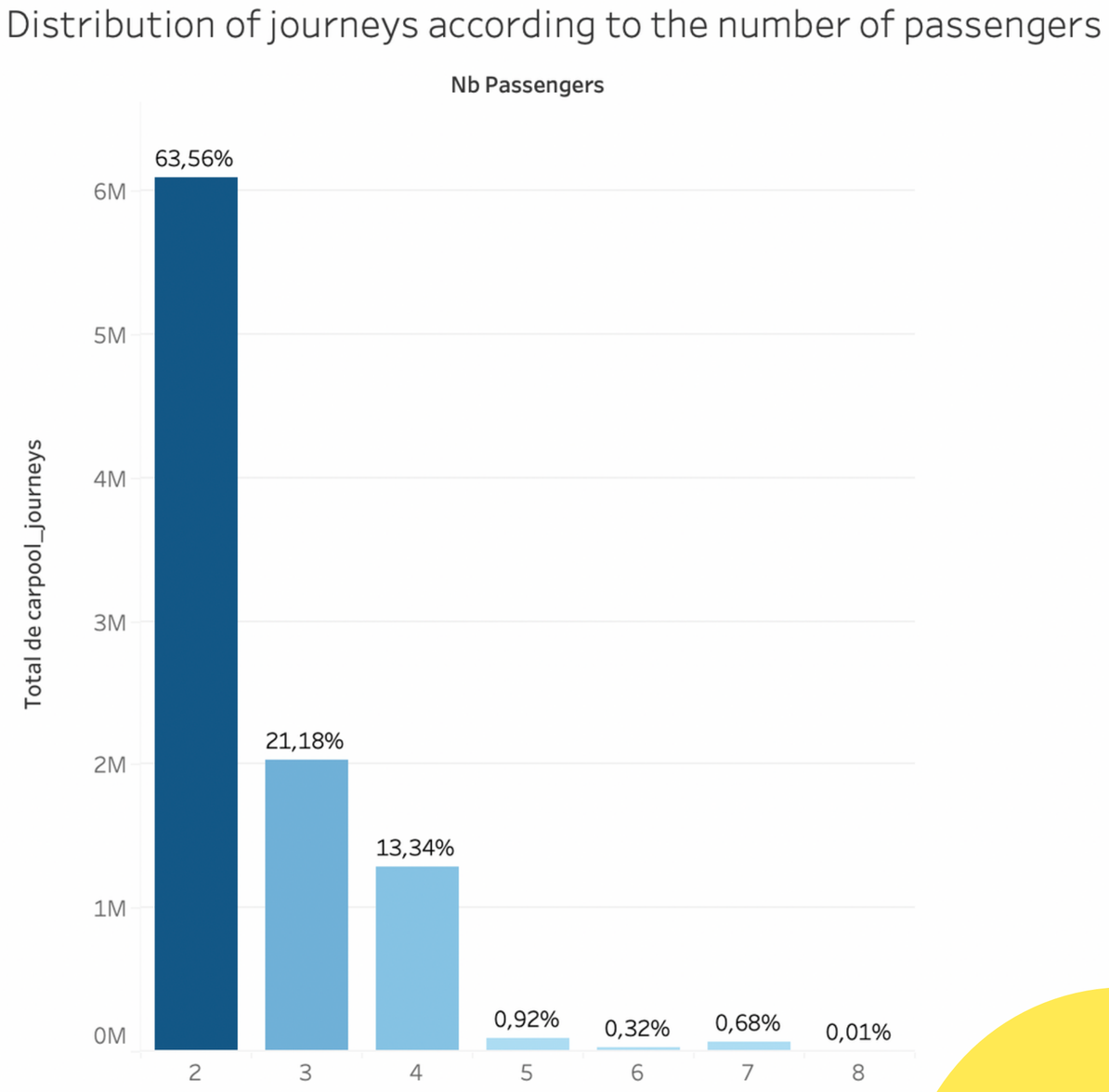
75% of the journeys are under 32km

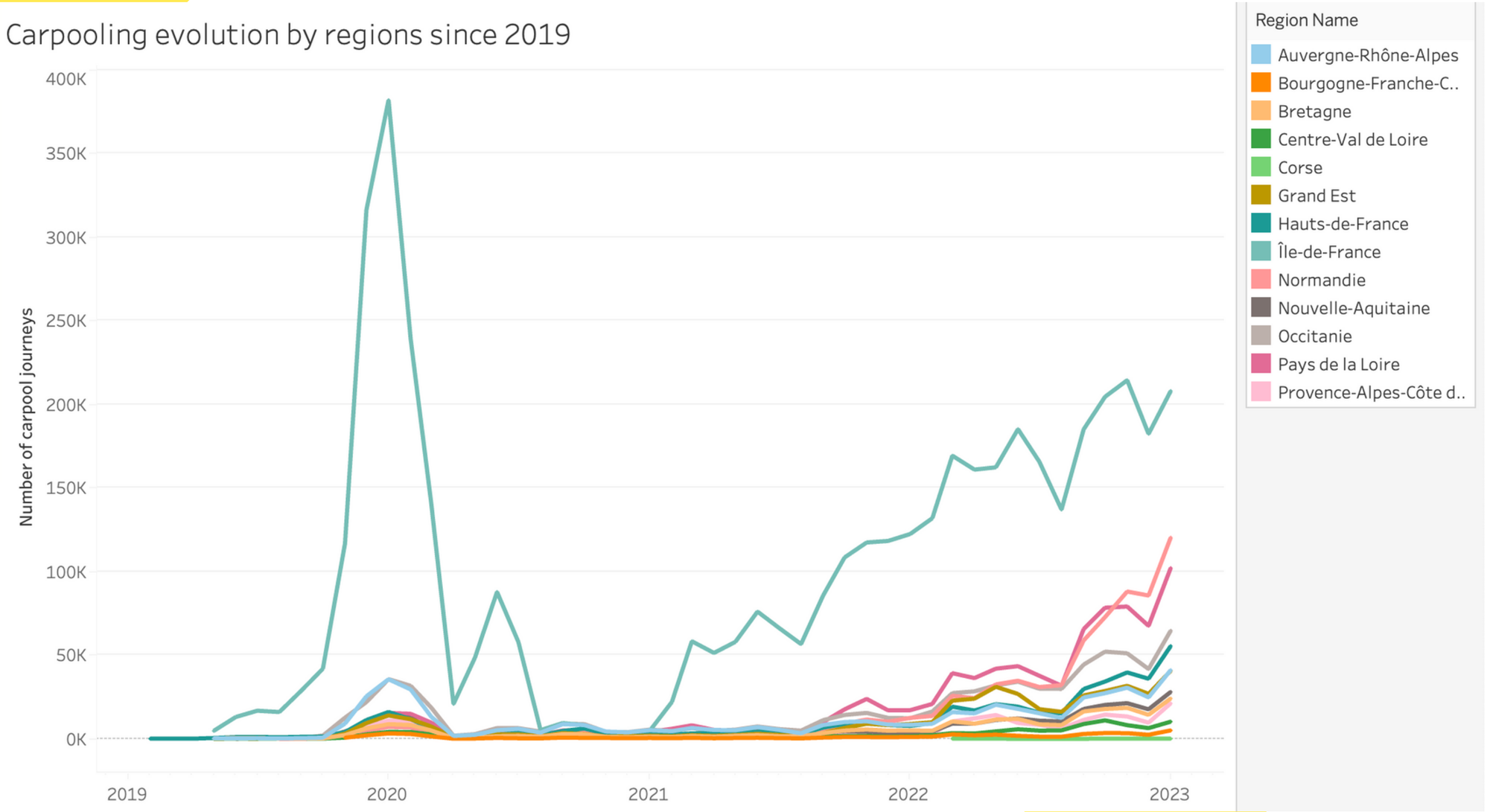




COMMUTING TRIPS

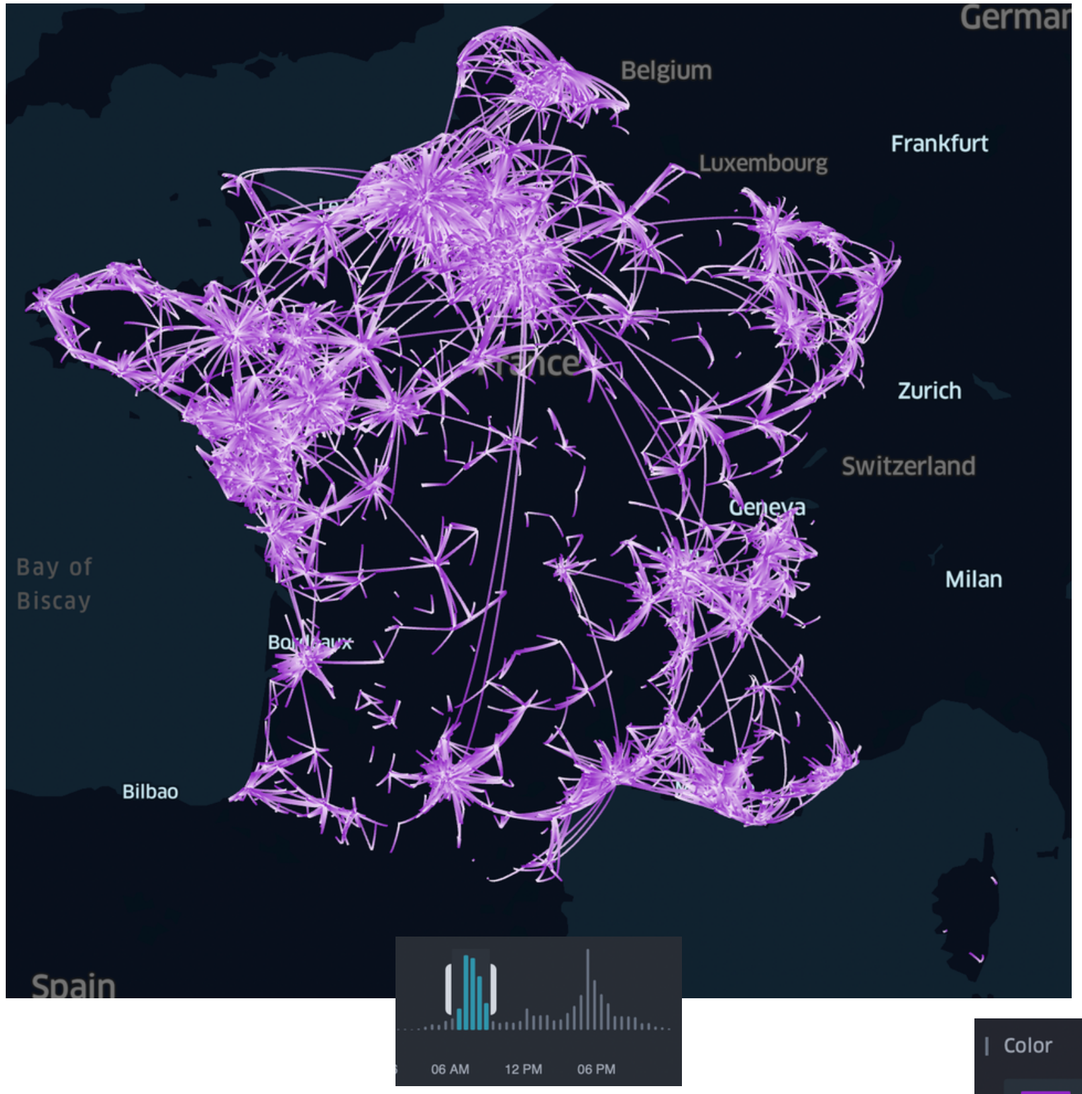
NUMBER OF PASSENGERS



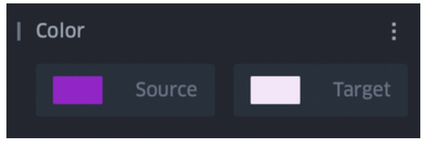


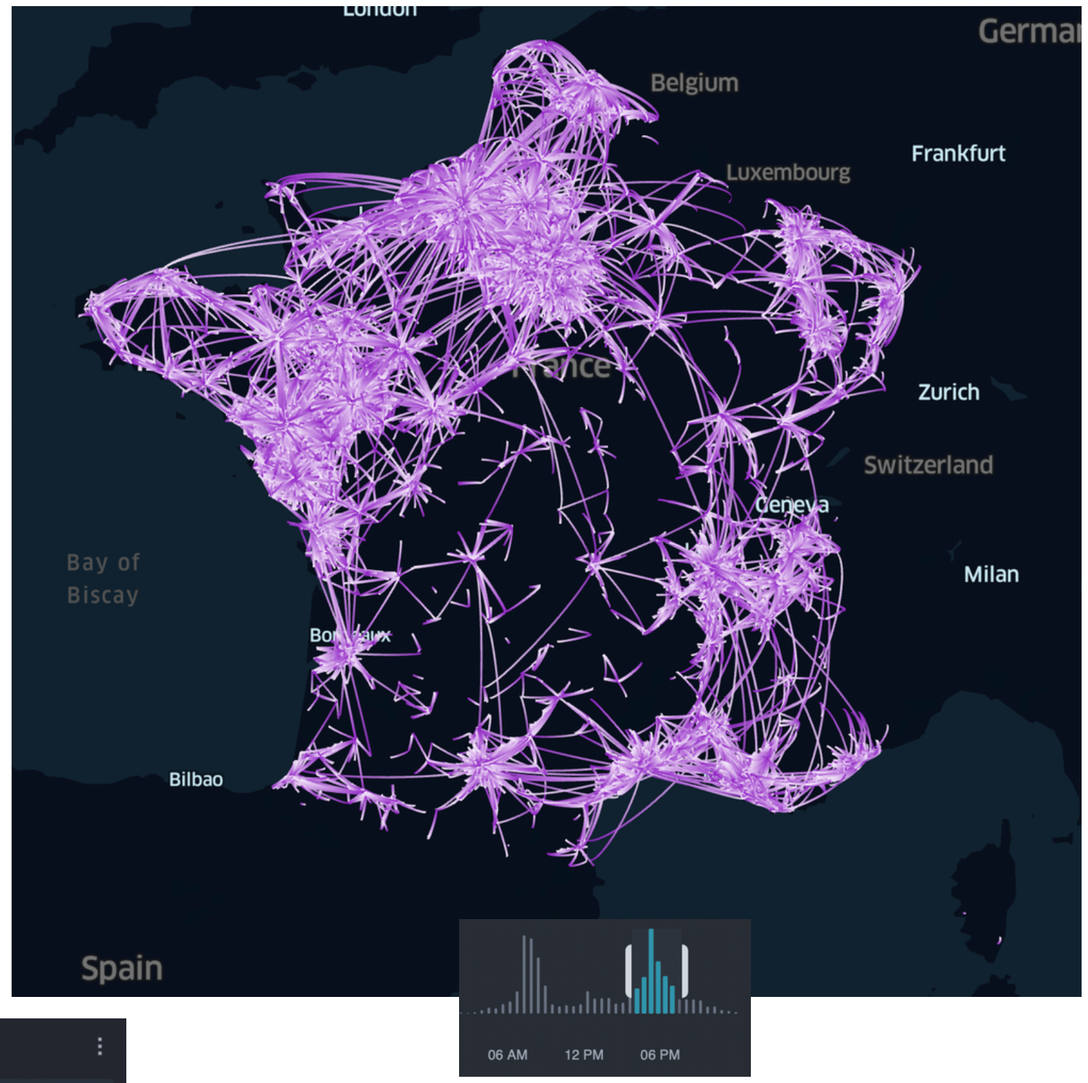
A REVIVAL OF CARPOOLING IN REGIONS

CARPOOL JOURNEYS



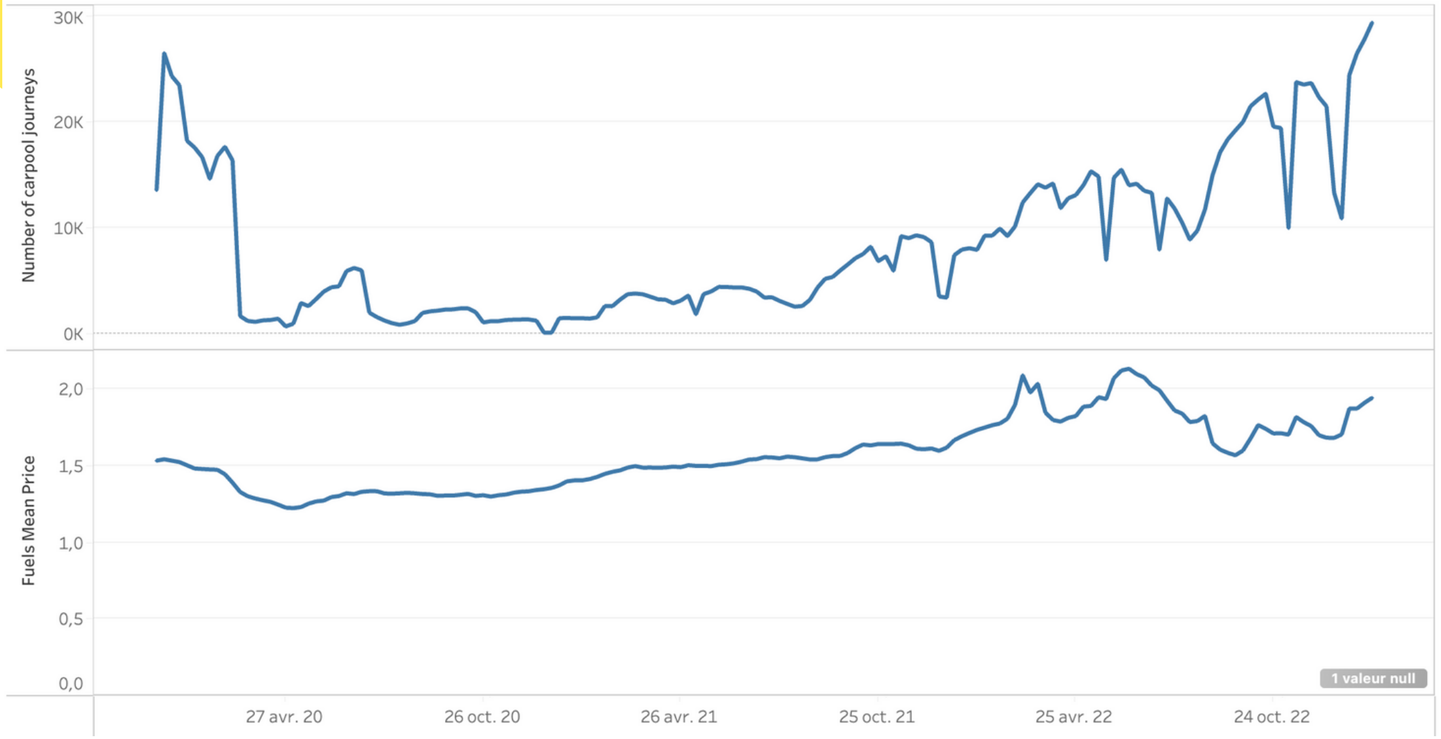
IN THE MORNING



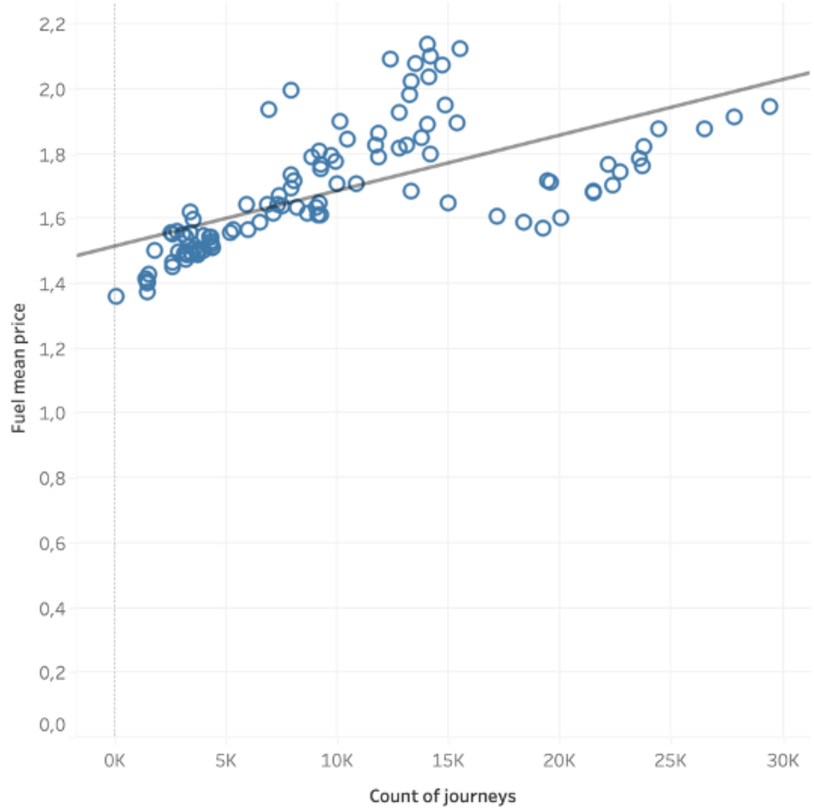


IN THE EVENING

Evolution of carpool journeys number and fuels mean price (weekly since 2020)

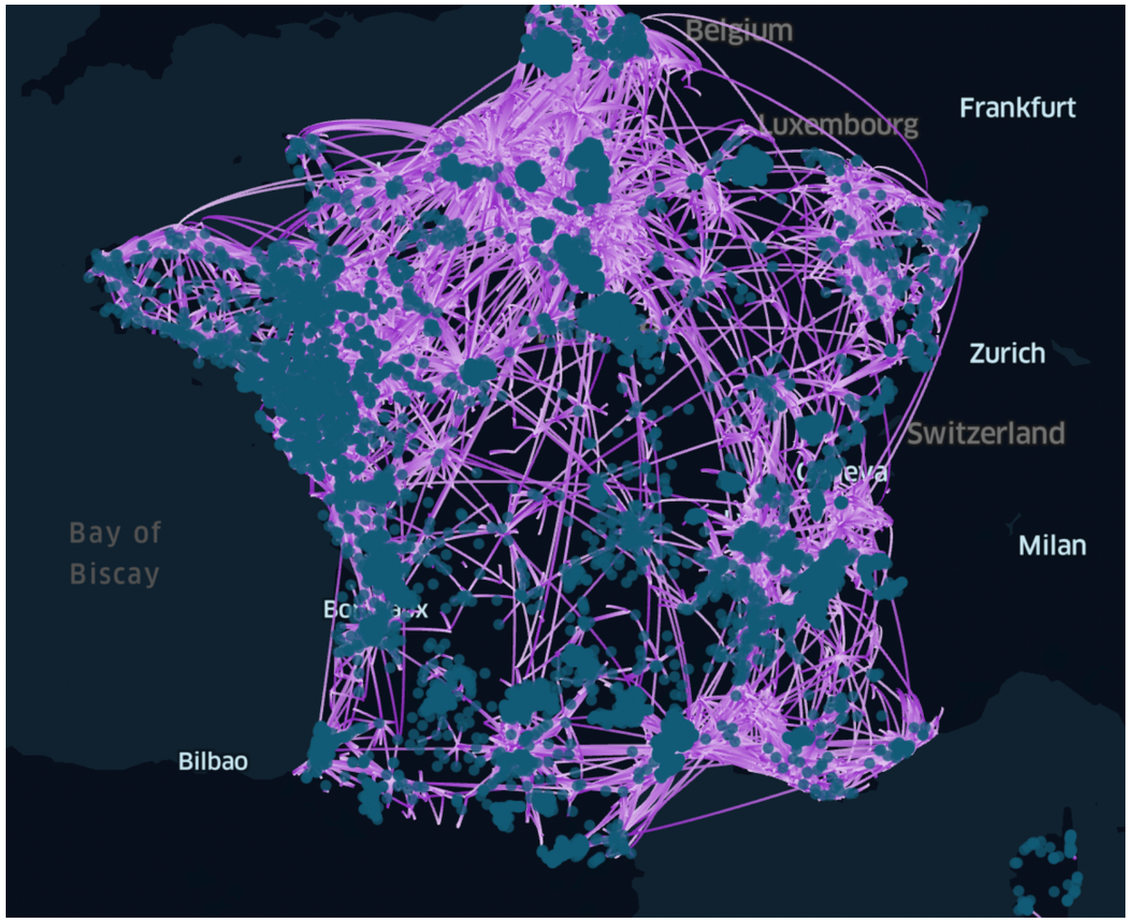


Correlation between fuel price and number of trips

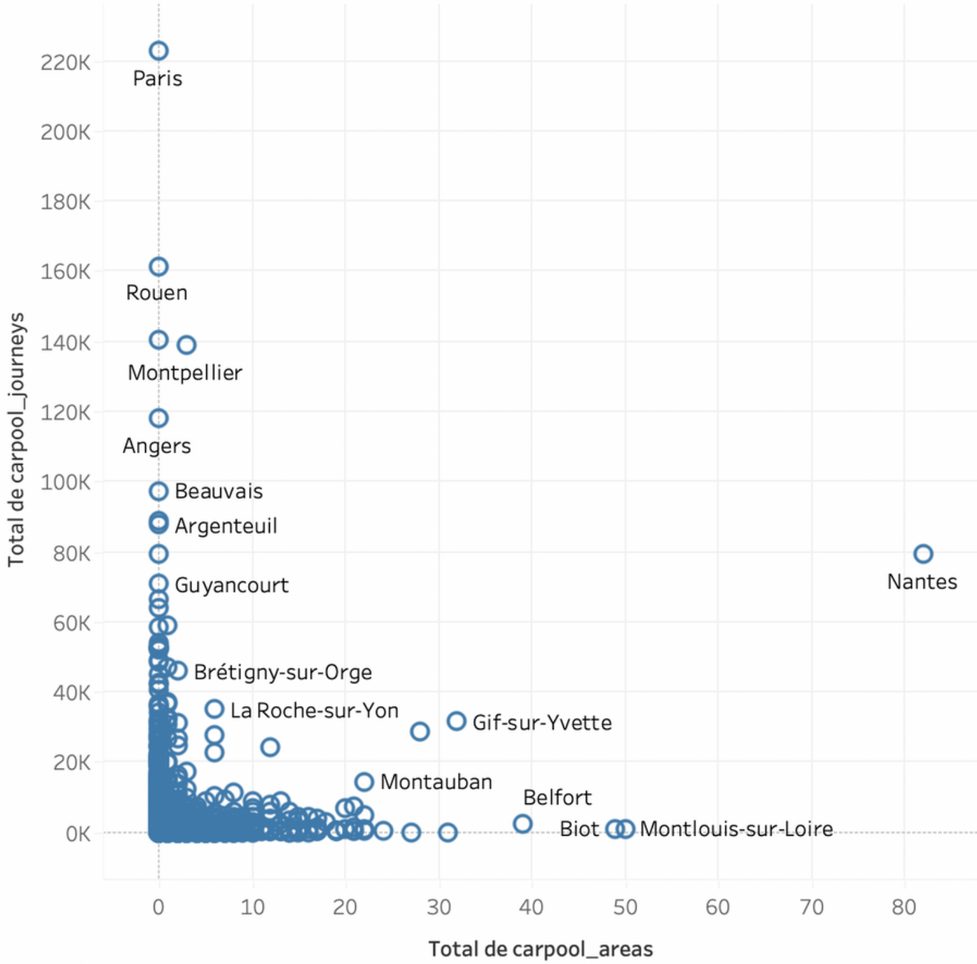


EXTERNAL CORRELATION : FUEL PRICE ?

Carpool parkings and journeys

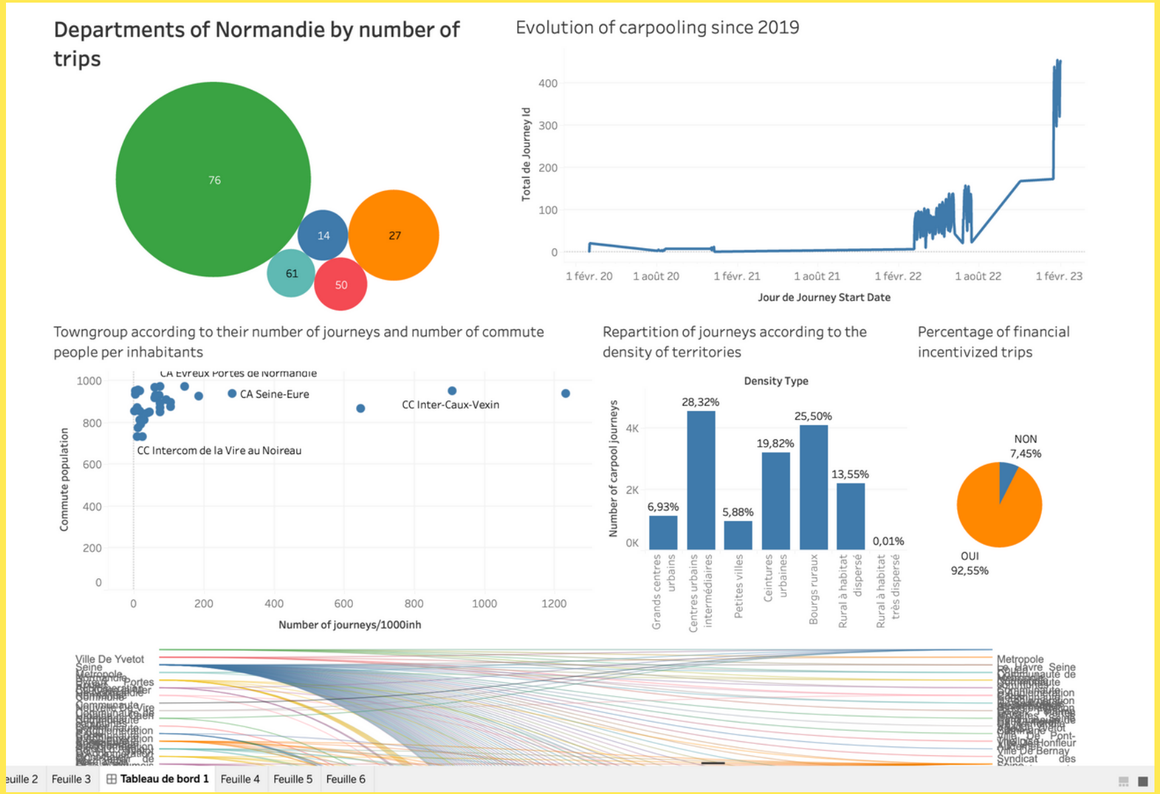


Relation between number of carpool parkings and number of carpool journeys



NEED A PARKING PLACE ?

FOLLOW YOUR TERRITORY ON A DASHBOARD



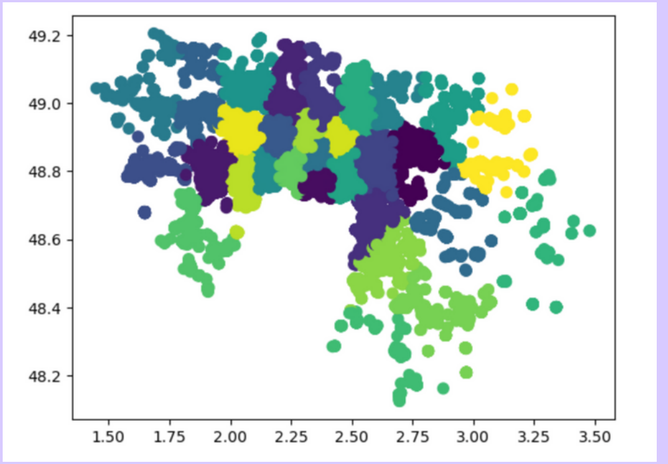
PREDICTION DEMO

How many carpool journeys can you find at your location and at this time ?

Where would this carpool go ?

MODELING PROCESS

Clustering



1 Clustering:

- K-means
- creation of the dataset with the number of journeys per cluster

```
clusters_df = df_2022_IDF[['journey_start_lat', 'journey_start_lon']]
```

```
from sklearn.cluster import KMeans
from sklearn import metrics
from scipy.spatial.distance import cdist
```

```
Y_axis = clusters_df[['journey_start_lat']]
X_axis = clusters_df[['journey_start_lon']]
```

```
import numpy as np
#Building and fitting the model
distortions = []
inertias = []
mapping1 = {}
mapping2 = {}
K = range(6,100)
X = clusters_df.sample(n=10000)

for k in K:
    kmeanModel = KMeans(n_clusters=k)
    kmeanModel.fit(X)

    distortions.append(sum(np.min(cdist(X, kmeanModel.cluster_centers_,
                      'euclidean'),axis=1)) / X.shape[0])
    inertias.append(kmeanModel.inertia_)

    mapping1[k] = sum(np.min(cdist(X, kmeanModel.cluster_centers_,
                  'euclidean'),axis=1)) / X.shape[0]
    mapping2[k] = kmeanModel.inertia_
```

journey_id	time	journey_start_lon	journey_start_lat	cluster
7745334	14:30:00	2.403	48.771	1
7745475	14:30:00	2.565	48.945	18
7721587	14:30:00	2.456	48.816	17
7721613	14:30:00	2.437	48.944	27
7721564	14:30:00	2.771	48.857	0

	cluster	time	nb_carpooling
0	0	00:00:00	20
1	0	00:10:00	27
2	0	00:20:00	23
3	0	00:30:00	14
4	0	00:40:00	19
...	...	...	...
3804	29	22:00:00	1
3805	29	22:10:00	1
3806	29	22:20:00	2
3807	29	22:50:00	1
3808	29	23:10:00	1

3809 rows × 3 columns

MODELING PROCESS

Clustering

KNN Regressor

2 Regression :

- Linear regression (bad idea)
- KNN Regressor (better idea)
- Hyperparameters
- Cross validation

```
knn2 = KNeighborsRegressor(**best_params)
knn2.fit(X_train_scaled, y_train)

y_train_pred=knn2.predict(X_train_scaled)
y_pred = knn2.predict(X_test_scaled)
```

```
Evaluation of the model on the training data
The root mean squared error is :  13.472598765673254
The R2 score is:  0.836962017598172
Evaluation of the model on the testing data
The root mean squared error is :  14.881844077145916
The R2 score is:  0.7723046519457891
```

```
Knn (first model)
Cross validation score : [ -444.1703937  -1071.4551706   -641.20498688 -1151.42845144
  -892.54822602]
Mean cross validation score : 840.1614457286138
Knn2 (best params)
Cross validation score : [ -459.06900016 -1043.79533711  -567.22250246 -1068.22078002
  -765.26398242]
Mean cross validation score : 780.7143204350195
```

MODELING PROCESS

Clustering

KNN Regressor

Decision Tree

3 Classification to predict the destination :

- Same clustering process but with the destination as target
- Decision Tree
- KNN Classifier

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

clf = DecisionTreeClassifier()
clf.fit(X_train, y_train)


y_pred = clf.predict(X_test)


accuracy = accuracy_score(y_test, y_pred)
print('Accuracy:', accuracy)
```

```
Accuracy: 0.6424242424242425
```

```
from sklearn.neighbors import KNeighborsClassifier
clf2 = KNeighborsClassifier(n_neighbors=5)
clf2.fit(X_train, y_train)


y_pred = clf2.predict(X_test)


accuracy = accuracy_score(y_test, y_pred)
print('Accuracy:', accuracy)
```

```
Accuracy: 0.5333333333333333
```

CHALLENGES AND IMPROVEMENTS

THE DATASET SIZE

For data cleaning
For Tableau and visualizations

DIMENSIONAL DATA

A lot of other data interesting : incomes, traffic, public transport offer, CO2 emissions

ML MODELS

Push the model on the destination guess by improving the parameters, do more explatory data to understand the clusters

CONCLUSIONS

Carpooling in France has suffered from Covid, but it is coming back in force with a greater diversity of regions involved.

The use of carpooling is extremely linked to commuting

Territories can analyse their data to see where the carpool could be encouraged.

Carpooling for commuting does not require infrastructure, but can be financially incentivized.

Carpooling needs to be established over time to create a habit. Also the size of cities doesn't determine the number of users, so maybe it depends more on operators and political will.

THANKS FOR YOUR ATTENTION

CLEMENCE LEGRAND

clemence.legrand@gmail.com

IRONHACK DATA ANALYST BOOTCAMP - PARIS MARCH 2023