



## Analyse des données du l'épidémie de coronavirus (Italie)

Licence sciences des données

Encadré par :

**Mohamed ELKahoui**

**Abdellah Massaq**

**Idir Ouasso**

Réalisé par :

**Yassin Latif**

**Mourad Ezzebdi**

## **Sommaire**

**Introduction générale**

**Partie 1 : Analyse du Projet**

**Objectives.....**

**Partie 2 : Réalisation**

**Importation bibliothèques des données.....**

**Nettoyage des données.....**

**Création de la base des données.....**

**Visualisation des données.....**

**Partie 3 : Etude statistique**

**Description Univariée.....**

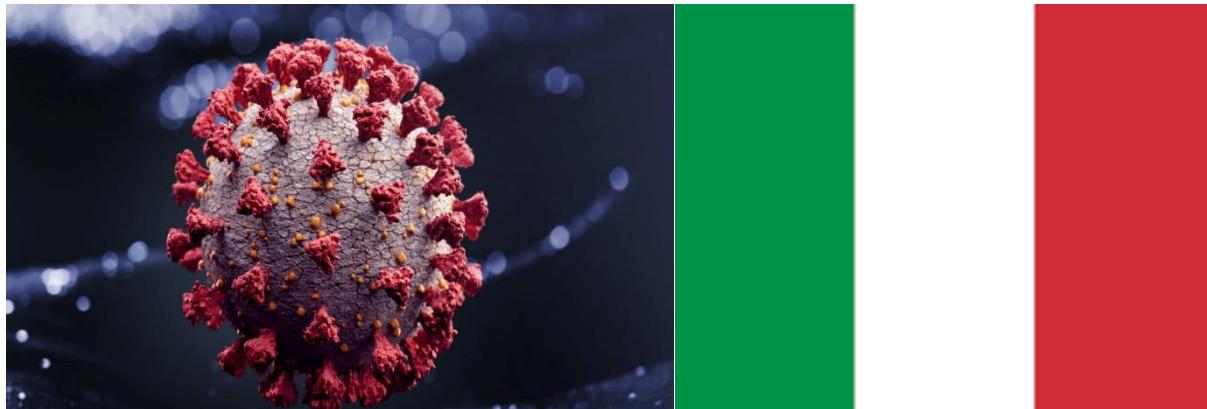
**Description Bivariée.....**

**Partie 4 : Conclusion**

**Partie 5 : Dashboard**

## Introduction générale

### Covid19 (l'Italie)



Les coronavirus (CoV) sont une grande famille de virus qui provoquent des maladies qui vont du simple rhume à des maladies plus graves telles que le syndrome respiratoire du Moyen-Orient (MERS-CoV) et le syndrome respiratoire aigu sévère (SRAS-CoV). Un nouveau coronavirus (nCoV) correspond à une nouvelle souche qui n'a pas été identifiée chez l'homme précédemment.

Les coronavirus sont de type zoonotique, c'est-à-dire qu'ils sont transmis de l'animal à l'homme. Des investigations détaillées ont révélé que le SRAS-CoV et le MERS-CoV étaient transmis à l'homme par les chats civettes et les dromadaires respectivement. Plusieurs coronavirus connus circulent chez des animaux qui n'ont pas encore infecté l'homme.

Les signes courants de l'infection sont les symptômes respiratoires, la fièvre, la toux, l'essoufflement et les difficultés respiratoires. Dans les cas les plus graves, l'infection peut provoquer une pneumonie, un syndrome respiratoire aigu sévère, une insuffisance rénale et même la mort.

Les recommandations standard pour prévenir la propagation de l'infection comprennent le lavage régulier des mains, le fait de se couvrir la bouche et le nez lorsqu'on tousse et éternue, la cuisson complète de la viande et des œufs. Éviter tout contact étroit avec toute personne présentant des symptômes de maladie respiratoire tels que la toux et les éternuements.

\*\*\*\*\*  
La situation sanitaire liée à l'épidémie de coronavirus continue de se dégrader dans le monde, notamment en France **en Italie**, incitant de nombreuses capitales à durcir leurs dispositifs dans l'espoir d'éloigner le spectre d'un deuxième vague à même de provoquer un nouveau séisme économique et social.

## Partie 1 : Analyse du projet

### Objectives :

Dans ce projet, nous allons essayer d'analyser les données de l'épidémie Covid-19 ou Sars-Cov-2 d'un pays qui est l'un des pays les plus touchés par Covid-19 qui est [l'Italie](#).

A fin de ce projet il faut atteindre plusieurs objectives :

- Création et manipulation d'une base de données.
- L'utilisation du langage SQL à travers sqlite3 ou MYSQL.
- Une étude statistique et exploitation des données représentées dans un dashboard.
- Apprenez à analyser vos données.

Méthode utilisée :

- Python (Jupyter)
- Sqlite3
- Power Bi
- R
- DB Browser

## Partie 2 : Réalisation

### Importation des Bibliothèques et données :

Premièrement on va importer les bibliothèques essentielles qu'on aura besoin sur la manipulation des données :

```

# essential libraries
import json
import random
from urllib.request import urlopen
import requests
import lxml.html as lh

# storing and analysis
import numpy as np
import pandas as pd

# visualization
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.graph_objs as go
import plotly.figure_factory as ff
import calmap
import folium
import seaborn as sns

# offline plotly visualization
from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot
import plotly as py
import plotly.graph_objs as go
init_notebook_mode(connected=True)

# color palette
tpc = '#393e46' # confirmed - grey
dth = '#ff2e63' # death - red
rec = '#21bf73' # recovered - cyan
act = '#fe9801' # active cases - yellow
hos = '#d2691e' # hospitalized cases - brown

# converter
from pandas.plotting import register_matplotlib_converters
register_matplotlib_converters()

```

Deuxièmement on va importer Les Données des deux 2 tables grâce à deux fichiers (.csv)

```

# importing datasets
Data_Byregion = pd.read_csv(r"/Users/yassinelelatif/Desktop/Project.lsd/covid19_italy_region.csv",
                            names = ['SNo', 'Date', 'Country', 'RegionCode', 'RegionName', 'Latitude', 'Longitude', 'Hospita
                            header = 0,
                            index_col = False)
Data_Byregion['Date'] = pd.to_datetime(Data_Byregion['Date'])
Data_Byregion.replace("Emilia Romagna", "Emilia-Romagna", inplace = True)

```

```

# importing datasets
Data_Byprovince = pd.read_csv(r"/Users/yassinelelatif/Desktop/Project.lsd/covid19_italy_province.csv", parse_dates=[ "Date"]
                             names = ['SNo', 'Date', 'Country', 'RegionCode', 'RegionName', 'ProvinceCode', 'ProvinceName', '
                             header = 0,
                             index_col = False)
Data_Byprovince.head()

```

Après l'importation des Bibliothèques et des données on fait la description des variables de chaque table :

### Description des variables :

#### Par région :

- SNo : Numéro de série
- Date : Date de notification au format AAAA-MM-JJTHH : MM : SS (ISO 8601)
- Country : Pays au format XYZ (ISO 3166-1 alpha-3)
- RegionCode : Code de la région (ISTAT 2019)
- RegionName : Nom de la région
- Latitude : Latitude par région
- Longitude : Longitude par région
- HospitalizedPatients : Patients hospitalisés présentant des symptômes, non en soins intensifs
- IntensiveCarePatients : Patients en soins intensifs

- TotalHospitalizedPatients : Total des patients hospitalisés (patients hospitalisés + patients en soins intensifs)
- HomeConfinement : Les personnes en quarantaine par confinement à domicile
- CurrentPositiveCases : Nombre total de cas positifs actuels (patients hospitalisés en quarantaine domestique)
- NewPositiveCases : Nouveau nombre de cas positifs actuels (HospitalizedPatients + HomeConfinement)
- Recovered : Nombre de cas récupérés
- Deaths : Nombre de décès
- TotalPositiveCases : Nombre total de cas positifs
- TestsPerformed : Nombre de tests effectués

### Par province :

- SNo : Numéro de série
- Date : Date de notification au format AAAA-MM-JJTHH : MM : SS (ISO 8601)
- Country : Pays au format XYZ (ISO 3166-1 alpha-3)
- RegionCode : Code de la région (ISTAT 2019)
- RegionName : Nom de la région
- ProvinceCode : Code de la province (ISTAT 2019)
- ProvinceName : Nom de la province
- ProvinceAbbreviation : Province abrégée (2 lettres)
- Latitude : Latitude par province
- Longitude : Longitude par province
- TotalPositiveCases : Nombre total de cas positifs par province

Nous allons prendre une première visualisation sur les deux tableaux :

### Première Table par région :

Data_Byregion.head()											
SNo	Date	Country	RegionCode	RegionName	Latitude	Longitude	HospitalizedPatients	IntensiveCarePatients	TotalHospitalizedPatients	HomeConfine	
0	2020-02-24 18:00:00	ITA	13	Abruzzo	42.351222	13.398438	0	0	0	0	
1	2020-02-24 18:00:00	ITA	17	Basilicata	40.639471	15.805148	0	0	0	0	
2	2020-02-24 18:00:00	ITA	18	Calabria	38.905976	16.594402	0	0	0	0	
3	2020-02-24 18:00:00	ITA	15	Campania	40.839566	14.250850	0	0	0	0	
4	2020-02-24 18:00:00	ITA	8	Emilia-Romagna	44.494367	11.341721	10	2	12		

### Deuxième Table par province :

```
Data_Byprovince.head()
```

SNo	Date	Country	RegionCode	RegionName	ProvinceCode	ProvinceName	ProvinceAbbreviation	Latitude	Longitude	TotalPositiveCases
0	0 2020-02-24 18:00:00	ITA	13	Abruzzo	66	L'Aquila	AQ	42.351222	13.398438	0
1	1 2020-02-24 18:00:00	ITA	13	Abruzzo	67	Teramo	TE	42.658918	13.704400	0
2	2 2020-02-24 18:00:00	ITA	13	Abruzzo	68	Pescara	PE	42.464584	14.213648	0
3	3 2020-02-24 18:00:00	ITA	13	Abruzzo	69	Chieti	CH	42.351032	14.167546	0
4	4 2020-02-24 18:00:00	ITA	13	Abruzzo	979	In fase di definizione/aggiornamento	NaN	NaN	NaN	0

Plus d'information sur les types des colonnes :

Pour la première table :

```
# dataframe info
Data_Byregion.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3969 entries, 0 to 3968
Data columns (total 17 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   SNo              3969 non-null    int64  
 1   Date             3969 non-null    datetime64[ns]
 2   Country          3969 non-null    object  
 3   RegionCode       3969 non-null    int64  
 4   RegionName       3969 non-null    object  
 5   Latitude          3969 non-null    float64 
 6   Longitude         3969 non-null    float64 
 7   HospitalizedPatients  3969 non-null    int64  
 8   IntensiveCarePatients  3969 non-null    int64  
 9   TotalHospitalizedPatients  3969 non-null    int64  
 10  HomeConfinement    3969 non-null    int64  
 11  CurrentPositiveCases  3969 non-null    int64  
 12  NewPositiveCases     3969 non-null    int64  
 13  Recovered          3969 non-null    int64  
 14  Deaths             3969 non-null    int64  
 15  TotalPositiveCases    3969 non-null    int64  
 16  TestsPerformed      2814 non-null    float64 
dtypes: datetime64[ns](1), float64(3), int64(11), object(2)
memory usage: 527.3+ KB
```

Et pour la deuxième table :

```
# dataframe info
Data_Byprovince.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25599 entries, 0 to 25598
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   SNo              25599 non-null    int64  
 1   Date             25599 non-null    datetime64[ns]
 2   Country          25599 non-null    object  
 3   RegionCode       25599 non-null    int64  
 4   RegionName       25599 non-null    object  
 5   ProvinceCode      25599 non-null    int64  
 6   ProvinceName      25599 non-null    object  
 7   ProvinceAbbreviation  20034 non-null    object  
 8   Latitude          20223 non-null    float64 
 9   Longitude         20223 non-null    float64 
 10  TotalPositiveCases  25599 non-null    int64  
dtypes: datetime64[ns](1), float64(2), int64(4), object(4)
memory usage: 2.1+ MB
```

## Nettoyage des données :

### Pour le premier tableau :

Détection des valeurs manquantes par deux méthodes :

#### Première Méthode :

```
# checking for missing value  
Data_Byregion.isna().sum()
```

SNo	0
Date	0
Country	0
RegionCode	0
RegionName	0
Latitude	0
Longitude	0
HospitalizedPatients	0
IntensiveCarePatients	0
TotalHospitalizedPatients	0
HomeConfinement	0
CurrentPositiveCases	0
NewPositiveCases	0
Recovered	0
Deaths	0
TotalPositiveCases	0
TestsPerformed	1155
	dtype: int64

Cette méthode donne la somme des valeurs manquantes dans chaque colonne et on observe dans ce cas qu'on a 1155 valeurs manquantes dans la colonne TestsPerformed.

#### Deuxième Méthode :

```
: Data_Byregion.isnull()
```

IntensiveCarePatients	TotalHospitalizedPatients	HomeConfinement	CurrentPositiveCases	NewPositiveCases	Recovered	Deaths	TotalPositiveCases	TestsPerformed
False	False	False	False	False	False	False	False	True
False	False	False	False	False	False	False	False	True
False	False	False	False	False	False	False	False	True
False	False	False	False	False	False	False	False	True
False	False	False	False	False	False	False	False	True
...	...	...	...	...	...	...	...	...
False	False	False	False	False	False	False	False	False
False	False	False	False	False	False	False	False	False
False	False	False	False	False	False	False	False	False
False	False	False	False	False	False	False	False	False

Dans cette méthode on voit des 'False' et des 'True', pour False veux dire qu'on a pas de valeur manquante mais par contre True veux dire qu'on des valeurs manquantes.

D'après les résultats, on remarque que la colonne 'TestsPerformed' contient 1155 valeurs manquantes.

Il existe plusieurs méthodes pour gérer les valeurs manquantes, mais dans notre cas nous allons les remplacer par la valeur entière de la médiane de notre colonne 'TestsPerformed' pour être plus logique car on peut pas faire un test-demi c'est pour ce là on a remplacé les valeurs manquantes par la valeur entière de la médiane.

```
median = int(Data_Byregion["TestsPerformed"].median())
Data_Byregion["TestsPerformed"].fillna(median, inplace=True)
```

### Vérification du nettoyage des valeurs manquantes :

```
|: Data_Byregion.isnull().sum()
SNo          0
Date         0
Country      0
RegionCode   0
RegionName   0
Latitude     0
Longitude    0
HospitalizedPatients  0
IntensiveCarePatients 0
TotalHospitalizedPatients 0
HomeConfinement 0
CurrentPositiveCases 0
NewPositiveCases 0
Recovered     0
Deaths        0
TotalPositiveCases 0
TestsPerformed 0
dtype: int64
```

Maintenant notre tableau ne contient aucune valeur manquante, on passe pour détecter les valeurs manquantes sur le deuxième tableau.

### Pour le deuxième tableau :

#### Détection des valeurs manquantes par deux méthodes :

##### Première Méthode :

```
|: Data_Byprovince.isnull().sum()
SNo          0
Date         0
Country      0
RegionCode   0
RegionName   0
ProvinceCode 0
ProvinceName 0
ProvinceAbbreviation 5565
Latitude     5376
Longitude    5376
TotalPositiveCases 0
dtype: int64
```

Cette méthode donne la somme des valeurs manquantes dans chaque colonne comme on voit au-dessus :

- Pour la colonne 'ProvinceAbbreviation' : on a 5565 valeurs manquantes.
- Pour la colonne 'Latitude' : on a 5376 valeurs manquantes.
- Pour la colonne 'Longitude' : on a 5376 valeurs manquantes.

Donc pour la sommes des valeurs manquantes de toute la tables on 16317 valeurs manquantes.

##### Deuxième Méthode :

Data_Byprovince.isnull()												
SNo	Date	Country	RegionCode	RegionName	ProvinceCode	ProvinceName	ProvinceAbbreviation	Latitude	Longitude	TotalPositiveCases		
0	False	False	False	False	False	False	False	False	False	False	False	
1	False	False	False	False	False	False	False	False	False	False	False	
2	False	False	False	False	False	False	False	False	False	False	False	
3	False	False	False	False	False	False	False	False	False	False	False	
4	False	False	False	False	False	False	False	True	True	True	False	
...	...	...	...	...	...	...	...	...	...	...	...	
25594	False	False	False	False	False	False	False	False	False	False	False	
25595	False	False	False	False	False	False	False	False	False	False	False	
25596	False	False	False	False	False	False	False	False	False	False	False	
25597	False	False	False	False	False	False	False	True	True	True	False	
25598	False	False	False	False	False	False	False	True	True	True	False	

25599 rows x 11 columns

Cette méthode détecte seulement les colonnes qui contiennent les valeurs manquantes

Dans notre cas les colonnes qui contiennent les valeurs manquantes sont :

- ‘ProvinceAbbreviation’
- ‘Latitude’
- ‘Longitude’

Car ces colonnes qui contiennent les True dans la table au-dessus

Il existe plusieurs méthodes pour gérer les valeurs manquantes, mais dans notre cas nous allons remplacer les valeurs manquantes de la colonne ‘Latitude’ par la médiane de notre colonne ‘Latitude’, puis nous allons les remplacer dans la colonne ‘Longitude’ par la médiane de notre colonne ‘Longitude’ , mais par contre dans la colonne ‘ProvinceAbbreviation’ on va remplacer « NAN » par « ID ».

```
median = Data_Byprovince["Latitude"].median()
Data_Byprovince["Latitude"].fillna(median, inplace=True)
```

```
median = Data_Byprovince["Longitude"].median()
Data_Byprovince["Longitude"].fillna(median, inplace=True)
```

```
Data_Byprovince["ProvinceAbbreviation"].fillna("ID", inplace=True)
```

Vérification du nettoyage des valeurs manquantes :

```
]: Data_Byprovince.isnull().sum()
```

SNo	0
Date	0
Country	0
RegionCode	0
RegionName	0
ProvinceCode	0
ProvinceName	0
ProvinceAbbreviation	0
Latitude	0
Longitude	0
TotalPositiveCases	0
dtype: int64	

## Création de la base des données :

Puisque qu'on a nettoyé les données des deux tables, on va créer une base de données pour stocker ces données nettoyées et on fait la connexion des deux tables nettoyées avec la base de données. ( en utilisant sqlite3 pour la création et la connexion et DB browser pour l'affichage et faire schéma relationnel ).

```
import sqlite3
conn = sqlite3.connect('Projet_Lsd_Covid_19.db')
c = conn.cursor()
```

## Connexion de la table par région avec la base des données :

```
Data_Byregion.to_sql('Covid_19_in_italy_Byregion', conn, schema=None, if_exists='fail', index=True, index_label=None)
```

## Connexion de la table par province avec la base des données :

```
Data_Byprovince.to_sql('Covid_19_in_italy_Byprovince', conn, schema=None, if_exists='fail', index=True, index_label=None)
```

Le premier tableau (par région) sur le DB browser :

The screenshot shows the DB Browser for SQLite interface. The title bar reads "DB Browser for SQLite - /Users/yassinelatif/Desktop". The main window displays a table titled "Covid\_19\_in\_italy\_Byregion". The table has columns: SNo, Date, Country, RegionCode, RegionName, and Latitude. The data consists of 18 rows, each representing a region in Italy with its corresponding code and latitude. The interface includes standard database navigation buttons at the bottom.

	SNo	Date	Country	RegionCode	RegionName	Latitude
5	4	2020-02-24 18:...	ITA	8	Emilia-Romagna	44.49436681
6	5	2020-02-24 18:...	ITA	6	Friuli Venezia Gi...	45.6494354
7	6	2020-02-24 18:...	ITA	12	Lazio	41.89277044
8	7	2020-02-24 18:...	ITA	7	Liguria	44.41149315
9	8	2020-02-24 18:...	ITA	3	Lombardia	45.46679409
10	9	2020-02-24 18:...	ITA	11	Marche	43.61675973
11	10	2020-02-24 18:...	ITA	14	Molise	41.55774754
12	11	2020-02-24 18:...	ITA	21	P.A. Bolzano	46.49933453
13	12	2020-02-24 18:...	ITA	22	P.A. Trento	46.06893511
14	13	2020-02-24 18:...	ITA	1	Piemonte	45.0732745
15	14	2020-02-24 18:...	ITA	16	Puglia	41.12559576
16	15	2020-02-24 18:...	ITA	20	Sardegna	39.21531192
17	16	2020-02-24 18:...	ITA	19	Sicilia	38.11569725
18	17	2020-02-24 18:...	ITA	9	Toscana	43.76023077

Le deuxième tableau (par province) sur le DB browser :

The screenshot shows the DB Browser for SQLite interface with the 'Covid\_19\_in\_italy' table selected. The table contains 14 rows of data, each representing a region in Italy. The columns are: Country, RegionCode, RegionName, ProvinceCode, ProvinceName, and ProvinceAbbreviation. The data includes regions like Abruzzo, Basilicata, Calabria, and L'Aquila, with their respective codes and abbreviations.

	Country	RegionCode	RegionName	ProvinceCode	ProvinceName	ProvinceAbbreviation
1	'A	13	Abruzzo	66	L'Aquila	AQ
2	'A	13	Abruzzo	67	Teramo	TE
3	'A	13	Abruzzo	68	Pescara	PE
4	'A	13	Abruzzo	69	Chieti	CH
5	'A	13	Abruzzo	879	Fuori Regione / ...	ID
6	'A	13	Abruzzo	979	In fase di definiz...	ID
7	'A	17	Basilicata	76	Potenza	PZ
8	'A	17	Basilicata	77	Matera	MT
9	'A	17	Basilicata	880	Fuori Regione / ...	ID
10	'A	17	Basilicata	980	In fase di definiz...	ID
11	'A	18	Calabria	78	Cosenza	CS
12	'A	18	Calabria	79	Catanzaro	CZ
13	'A	18	Calabria	80	Reggio di Calabria	RC
14	'A	18	Calabria	101	Crotone	KR

Pour le schéma relationnel on observe que la variable commune entre les deux tables et la clé primaire de l'un des tables est 'RegionCode' qui est la clé primaire de la table par région donc c'est une clé étrangère de la table par province

Donc Le schéma Relationnel sera la suivante :

Covid\_19\_in\_italy\_Byregion (RegionCode, SNo, Date, Country, RegionName, Latitude, Longitude, HospitalizedPatient, IntensiveCarePatients, HomeConfiment, CurrentPositiveCases, NewPositiveCases, TotalHospitalizedPatients, Recoreved, Deaths, TotalPositiveCases, TestPerformed, Active)

Covid\_19\_in\_italy\_Byprovince (ProvinceCode, SNo, Date, Country, RegionName, ProvinceName, ProvinceAbbreviation, Latitude, Longitude, TotalPositiveCases, #RegionCode)

## Regroupement Des Données :

On va faire le regroupement de chaque table pour aider-nous pour faire la visualisation des données d'une manière logique plus clair

### Première Table (Par Région) :

```

# latest
data = Data_Byregion[Data_Byregion['Date'] == max(Data_Byregion['Date'])].reset_index()

# latest condensed
data_grouped = data.groupby('RegionName')['TotalPositiveCases', 'Deaths', 'Recovered', 'Active'].sum().reset_index()

#latest condensed with data about swabs (tests), quarantine and hospitalization
data_grouped_moreinfo = data.groupby('RegionName')[ 'TotalPositiveCases', 'Deaths', 'Recovered', 'Active','TestsPerformed','HomeConfined']

#Regional visualization adjustment (Merging Trento and Bolzano into Trentino-Alto Adige)
dgm_2 = data.copy()
dgm_2.replace("P.A. Bolzano", "Trentino-Alto Adige", inplace = True)
dgm_2.replace("P.A. Trento", "Trentino-Alto Adige", inplace = True)
dgm_2 = dgm_2.groupby('RegionName')[ 'TotalPositiveCases', 'Deaths', 'Recovered', 'Active','TestsPerformed','HomeConfined']

```

## Deuxième Table (Par Province) :

```

# latest
Data_Byprovince = Data_Byprovince[Data_Byprovince['Date'] == max(Data_Byprovince['Date'])].reset_index()

# latest condensed
data_grouped_province = Data_Byprovince.groupby('ProvinceName')[ 'TotalPositiveCases'].sum().reset_index()

```

## Création d'une colonne :

On va créer une colonne qui s'appelle 'Active' pour faire une visualisation générale des données

```

# cases
cases = [ 'TotalPositiveCases', 'Deaths', 'Recovered', 'Active']

# Active Case = confirmed - deaths - recovered
Data_Byregion['Active'] = Data_Byregion[ 'TotalPositiveCases'] - Data_Byregion[ 'Deaths'] - Data_Byregion[ 'Recovered']

```

Et maintenant on passe pour faire des analyses visuelles.

## Visualisation des données :

### Table 1 : (Par région)

Premièrement on va faire un visualisation sur les région les plus touchées par Covid-19 sur la table :

```
#Visualisation de les pays les plus touchées par COVID-19
temp = data.groupby(['RegionName'])['TotalPositiveCases', 'Deaths', 'Recovered', 'Active'].max()
temp.style.background_gradient(cmap='Reds')
```

RegionName	TotalPositiveCases	Deaths	Recovered	Active
Abruzzo	3773	472	2872	429
Basilicata	524	28	408	88
Calabria	1477	97	1148	232
Campania	6882	445	4412	2025
Emilia-Romagna	31805	4459	24478	2868
Friuli Venezia Giulia	3764	348	3056	360
Lazio	11043	878	7130	3035
Liguria	10907	1571	8827	509
Lombardia	99940	16863	76248	6829
Marche	7238	987	5949	302
Molise	525	23	432	70
P.A. Bolzano	2932	292	2450	190
P.A. Trento	5092	405	4600	87
Piemonte	32844	4146	27293	1405
Puglia	5402	556	4029	817
Sardegna	2114	134	1268	712
Sicilia	4291	286	2891	1114
Toscana	11785	1141	9141	1503
Umbria	1784	80	1442	262
Valle d'Aosta	1232	146	1063	23
Veneto	22864	2120	19399	1345

On observe D'après la table que les régions les plus touchées par Covid-19 sont :

- Lombardia : avec 99940 de Total des cas positive
- Piemonte : avec 32844 de Total des cas positive
- Emilia-Romagna : avec 31805 de Total des cas positive
- Veneto : avec 22864 de Total des cas positive

**Deuxièmement faire visualisation national sur Italie :**

```
temp = data.groupby('Date')['TotalPositiveCases', 'Deaths', 'Recovered', 'Active'].sum().reset_index()
temp = temp[temp['Date'] == max(temp['Date'])].reset_index(drop=True)
temp.style.background_gradient(cmap='Pastell')
```

Date	TotalPositiveCases	Deaths	Recovered	Active
0 2020-08-30 17:00:00	268218	35477	208536	24205

Donc on déduit qu'Italie a :

- 268218 de nombre de Total des cas positive.
- 35477 de nombre des Décès
- 208536 de nombre de personnes récupérées
- 24205 de nombre de personnes qui sont active

**Troisièmement On va faire la visualisation sur les régions plus touchées par Covid-19 par rapport au Décès par ordre Décroissante :**

```
: temp_dg = temp_f[temp_f['Deaths']>0][['RegionName', 'Deaths']]
temp_dg.sort_values('Deaths', ascending=False).reset_index(drop=True).style.background_gradient(cmap='Reds')
```

	RegionName	Deaths
0	Lombardia	16863
1	Emilia-Romagna	4459
2	Piemonte	4146
3	Veneto	2120
4	Liguria	1571
5	Toscana	1141
6	Marche	987
7	Lazio	878
8	Puglia	556
9	Abruzzo	472
10	Campania	445
11	P.A. Trento	405
12	Friuli Venezia Giulia	348
13	P.A. Bolzano	292
14	Sicilia	286
15	Valle d'Aosta	146
16	Sardegna	134
17	Calabria	97
18	Umbria	80
19	Basilicata	28
20	Molise	23

On observe que les régions les plus touchées par Covid-19 par rapport au Décès ce sont :

- Lombardia : avec 16863 de nombre des personnes décès.
- Emilia-Romagna : avec 4459 de nombre des personnes décès.
- Piemonte : avec 4146 de nombre des personnes décès.
- Veneto : avec 2120 de nombre des personnes décès.

**Visualisation si on a des régions sans cas signalé comme guérie :**

```
: temp = temp_f[temp_f['Recovered']==0][['RegionName', 'TotalPositiveCases', 'Deaths', 'Recovered']]
temp.reset_index(drop=True).style.background_gradient(cmap='Reds')
```

RegionName	TotalPositiveCases	Deaths	Recovered
------------	--------------------	--------	-----------

On déduit qu'on a aucune région sans cas signalé comme guéri

**Quatrièmement On va faire la visualisation sur les régions qu'ont plus de nombre de Guérisons par ordre Décroissante :**

```
: temp_dg = temp_f[temp_f['Deaths'] > 0][['RegionName', 'Recovered']]
temp_dg.sort_values('Recovered', ascending=False).reset_index(drop=True).style.background_gradient(cmap='Greens')
```

	RegionName	Recovered
0	Lombardia	76248
1	Piemonte	27293
2	Emilia-Romagna	24478
3	Veneto	19399
4	Toscana	9141
5	Liguria	8827
6	Lazio	7130
7	Marche	5949
8	P.A. Trento	4600
9	Campania	4412
10	Puglia	4029
11	Friuli Venezia Giulia	3056
12	Sicilia	2891
13	Abruzzo	2872
14	P.A. Bolzano	2450
15	Umbria	1442
16	Sardegna	1268
17	Calabria	1148
18	Valle d'Aosta	1063
19	Molise	432
20	Basilicata	408

On observe que les régions qu'ont les plus nombres des cas des Guérisons ce sont :

- Lombardia : avec 76248 de nombre des personnes guéries.
- Piemonte : avec 27293 de nombre des personnes guéries.
- Emilia-Romagna : avec 24478 de nombre des personnes guéries.
- Veneto : avec 19399 de nombre des personnes guéries.

**Visualisation les régions où les cas ne sont plus concernés :**

```
: temp = data_grouped[data_grouped['TotalPositiveCases'] ==
                     data_grouped['Deaths'] +
                     data_grouped['Recovered']]
temp = temp[['RegionName', 'TotalPositiveCases', 'Deaths', 'Recovered']]
temp = temp.sort_values('TotalPositiveCases', ascending=False)
temp = temp.reset_index(drop=True)
temp.style.background_gradient(cmap='Greens')
```

RegionName	TotalPositiveCases	Deaths	Recovered
------------	--------------------	--------	-----------

- On déduit qu'on a aucune région où les cas ne sont plus concernés

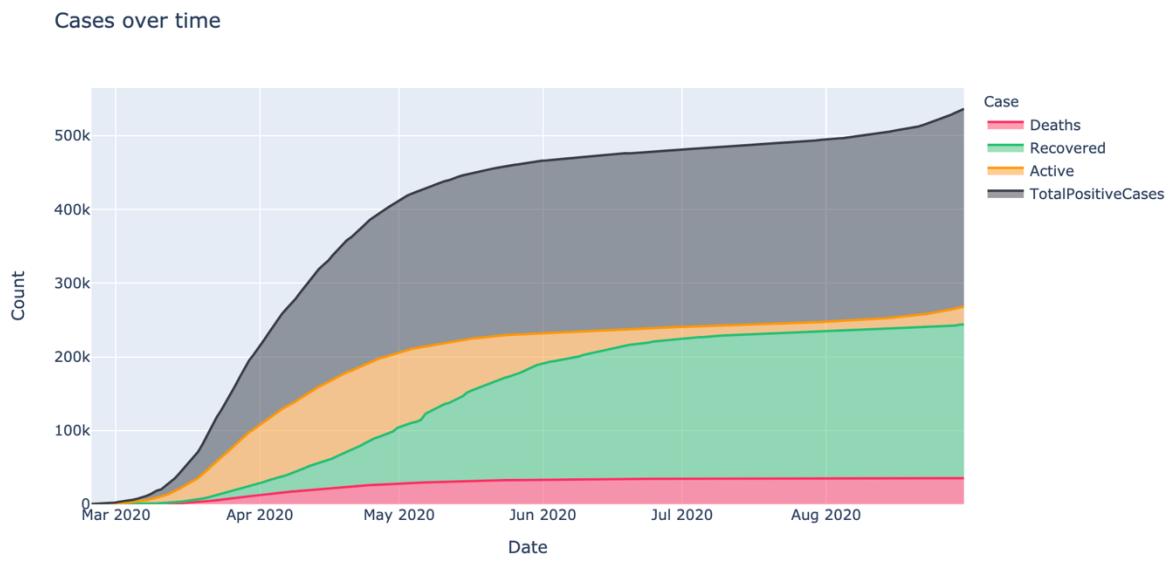
**Comparaison entre le total des cas positives, les cas actives, les cas de décès et les cas de guérisons :**

```

: temp = Data_Byregion.groupby('Date')[['Deaths', 'Recovered', 'Active','TotalPositiveCases']].sum().reset_index()
temp = temp.melt(id_vars="Date", value_vars=['Deaths', 'Recovered', 'Active', 'TotalPositiveCases'],
                 var_name='Case', value_name='Count')
temp.head()

fig = px.area(temp, x="Date", y="Count", color='Case',
              title='Cases over time', color_discrete_sequence = [dth, rec, act, tpc])
fig.show()

```



Dans ce graphe on constate que le nombre de décès augmente faiblement par contre le nombre des personnes récupérées sont de plus en plus nombreuses aussi que le nombre des cas actives augmente également et on note aussi que le nombre total des cas positive augment plus vite que les autres.

**Remarque :**

Notez s'il vous plaît :

Il est très probable que les taux indiqués ci-dessous surestiment la létalité réelle du COVID-19, car le nombre réel de personnes infectées pourrait facilement être supérieur aux cas confirmés.

**Visualisation De Taux de récupération, de mortalité et d'hospitalisation au fil du temps :**

```

temp = Data_Byregion.groupby('Date').sum().reset_index()

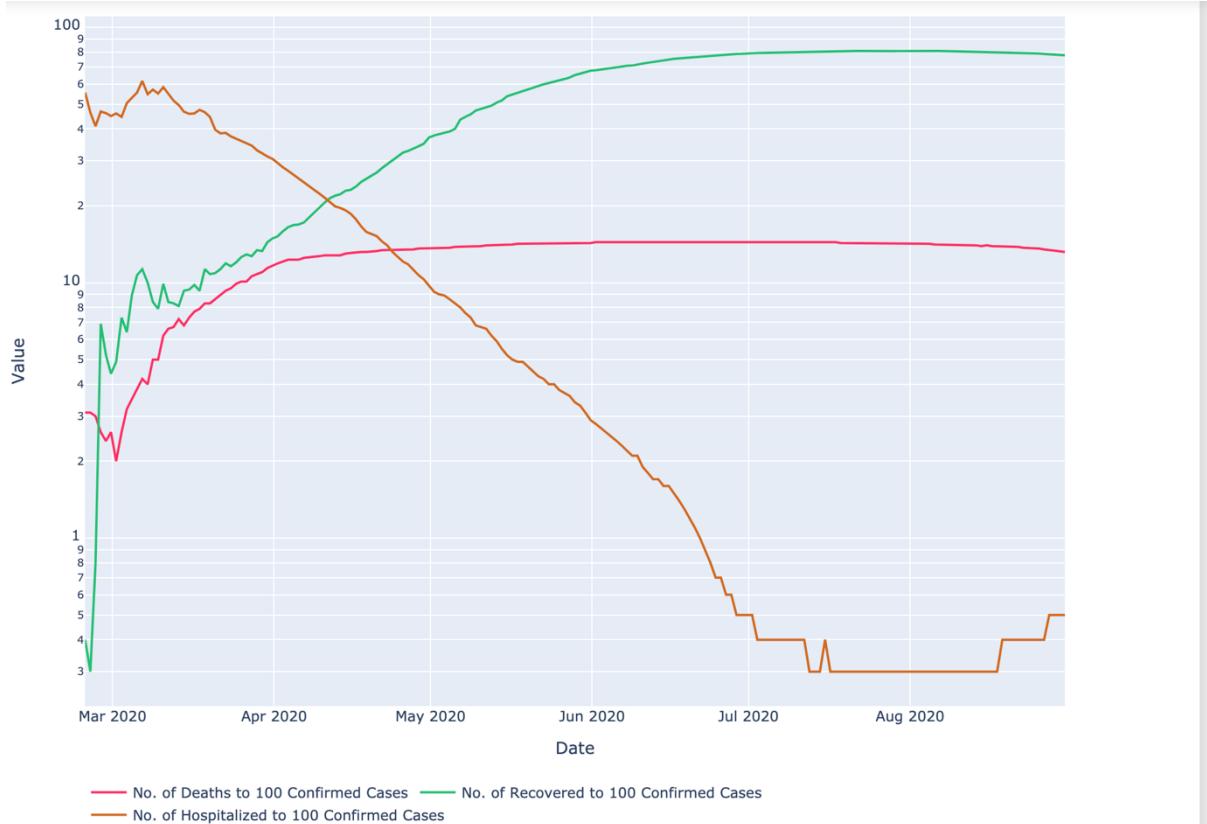
# adding two more columns
temp['No. of Deaths to 100 Confirmed Cases'] = round(temp['Deaths']/temp['TotalPositiveCases'], 3)*100
temp['No. of Recovered to 100 Confirmed Cases'] = round(temp['Recovered']/temp['TotalPositiveCases'], 3)*100
temp['No. of Hospitalized to 100 Confirmed Cases'] = round(temp['TotalHospitalizedPatients']/temp['TotalPositiveCase'])

# temp['No. of Recovered to 1 Death Case'] = round(temp['Recovered']/temp['Deaths'], 3)

temp = temp.melt(id_vars='Date', value_vars=['No. of Deaths to 100 Confirmed Cases', 'No. of Recovered to 100 Confirmed Cases'],
                 var_name='Ratio', value_name='Value')

fig = px.line(temp, x="Date", y="Value", color='Ratio', log_y=True,
              title='Recovery, Mortality and Hospitalization Rate Over The Time', color_discrete_sequence=[dth, rec],
              height=800)
fig.update_layout(legend_orientation="h", legend_title='')
fig.show()

```



D'après le graphe au-dessus :

- On note que le taux de récupération augmente vite que le taux de mortalité car au premier mars on observe que le taux de mortalité est plus grand que le taux de récupérations par contre après 10 jours on voit que le taux de récupérations plus grand que le taux de mortalités.
- On note aussi la décroissance de taux d'hospitalisation à cause la croissance de le taux de mortalité.
- Donc on déduit que l'Italie maintenant est stable à cause de l'augmentation vite du taux de récupérations et l'augmentation faible de le taux de mortalité.

## Visualisation du nombre de régions dans lesquelles le COVID-19 s'est propagé :

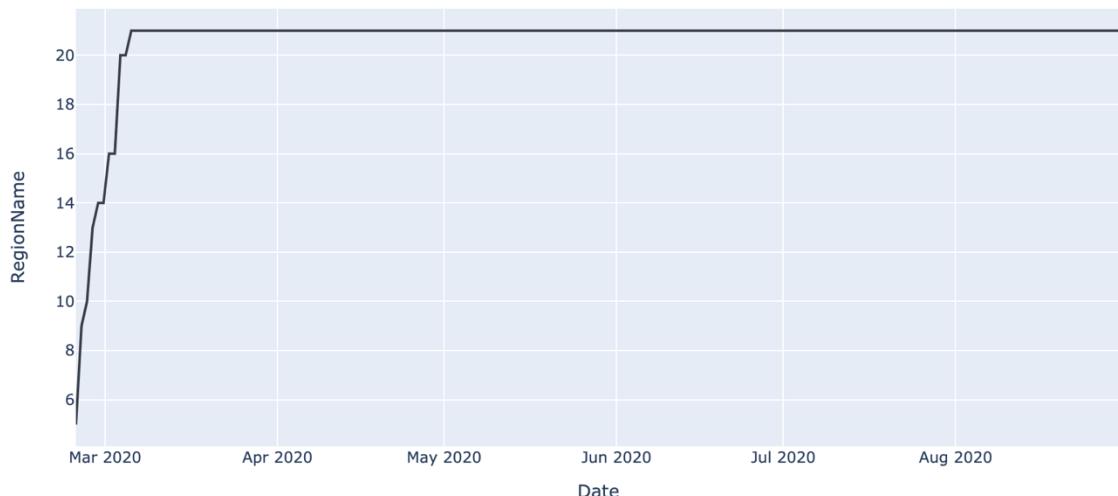
**Remarque :**

Comme mentionné précédemment, les provinces autonomes de Trente et Bolzano sont étiquetées comme des régions, de sorte que le nombre total de régions s'élève à 21.

```
: reg_spread = Data_Byregion[Data_Byregion['TotalPositiveCases']!=0].groupby('Date')[['RegionName']].unique().apply(len)
reg_spread = pd.DataFrame(reg_spread).reset_index()

fig = px.line(reg_spread, x='Date', y='RegionName',
              title='Number of Italian Regions to which COVID-19 spread over the time',
              color_discrete_sequence=[tpc,dth, rec])
fig.update_traces(textposition='top center')
fig.update_layout(uniformtext_minsize=5, uniformtext_mode='hide')
fig.show()
```

Number of Italian Regions to which COVID-19 spread over the time



- On note que Covid-19 s'est propagé dans 21 régions dans l'Italie
- Donc on déduit que covid-19 s'est propagé dans tout l'Italie

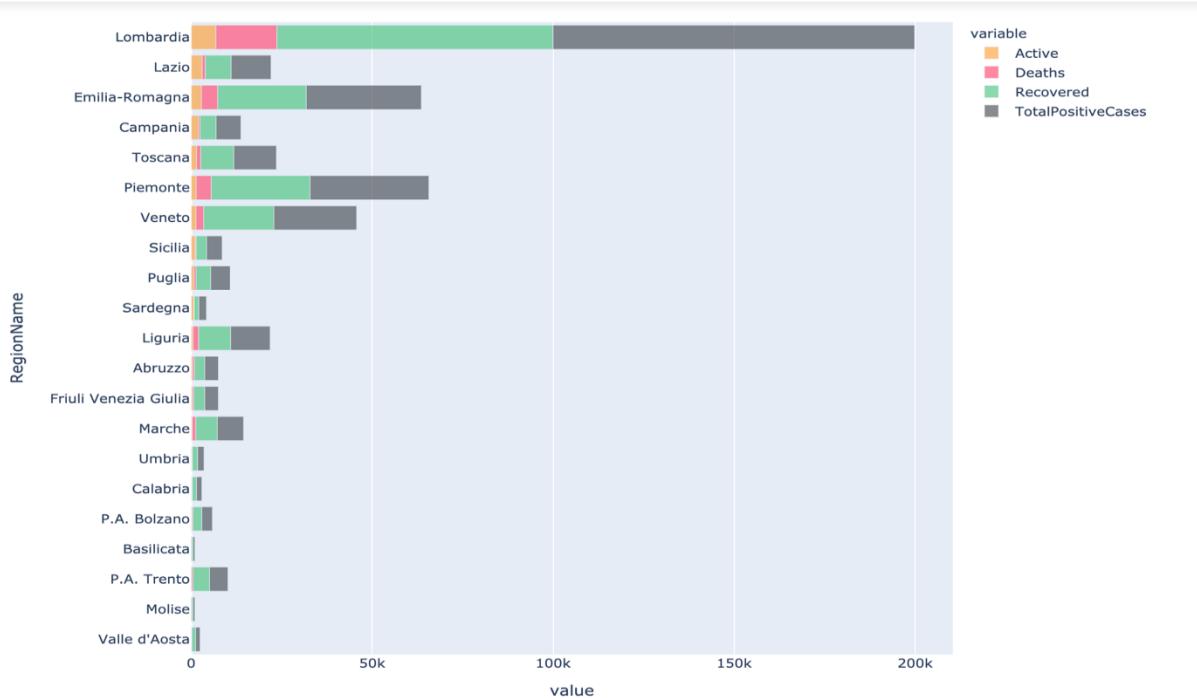
## La vue d'ensemble des cas par région

```
): cl = data.groupby('RegionName')['TotalPositiveCases', 'Deaths', 'Recovered'].sum()
cl = cl.reset_index().sort_values(by='TotalPositiveCases', ascending=False).reset_index(drop=True)
cl.head().style.background_gradient(cmap='rainbow')
```

	RegionName	TotalPositiveCases	Deaths	Recovered
0	Lombardia	99940	16863	76248
1	Piemonte	32844	4146	27293
2	Emilia-Romagna	31805	4459	24478
3	Veneto	22864	2120	19399
4	Toscana	11785	1141	9141

```
): ncl = cl.copy()
ncl['Active'] = ncl['TotalPositiveCases'] - ncl['Deaths'] - ncl['Recovered']
ncl = ncl.melt(id_vars="RegionName", value_vars=['Active', 'Recovered', 'Deaths', 'TotalPositiveCases'])

fig = px.bar(ncl.sort_values(['variable', 'value']),
             y="RegionName", x="value", color='variable', orientation='h', height=800,
             title='Number and state of Cases by Region', color_discrete_sequence=[act, dth, rec, tpc])
fig.update_layout(uniformtext_minsize=8, uniformtext_mode='hide')
fig.update_traces(opacity=0.6)
fig.show()
```

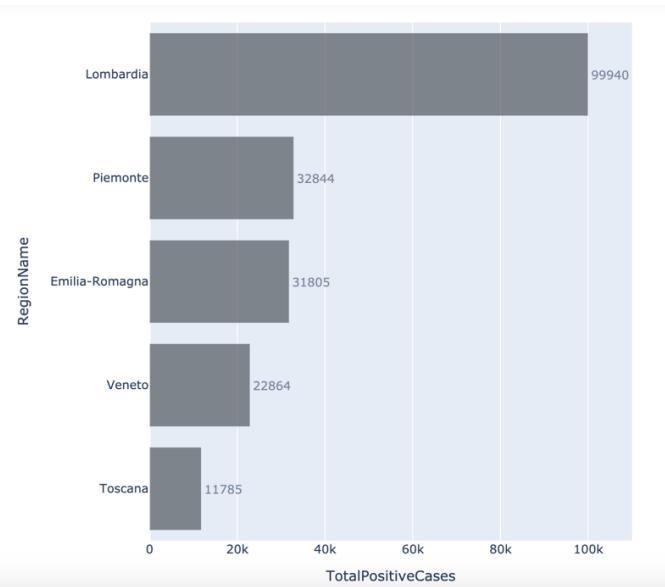


- On note que les régions suivantes : Lombardia , Emilia-Romagna , Piemonte et Veneto et Liguria sont les régions les plus touchées par Covid-19.
- Et on note aussi que les région suivantes : Lazio, Valle d'Aosta, Molise, P.A. Trento, Umbria, Calaria, Basilicata, Friuli Venezia Giulia les moins touchées par Covid-19.

## Top 5 des régions par catégorie :

### 1. Par le Nombre des cas positive :

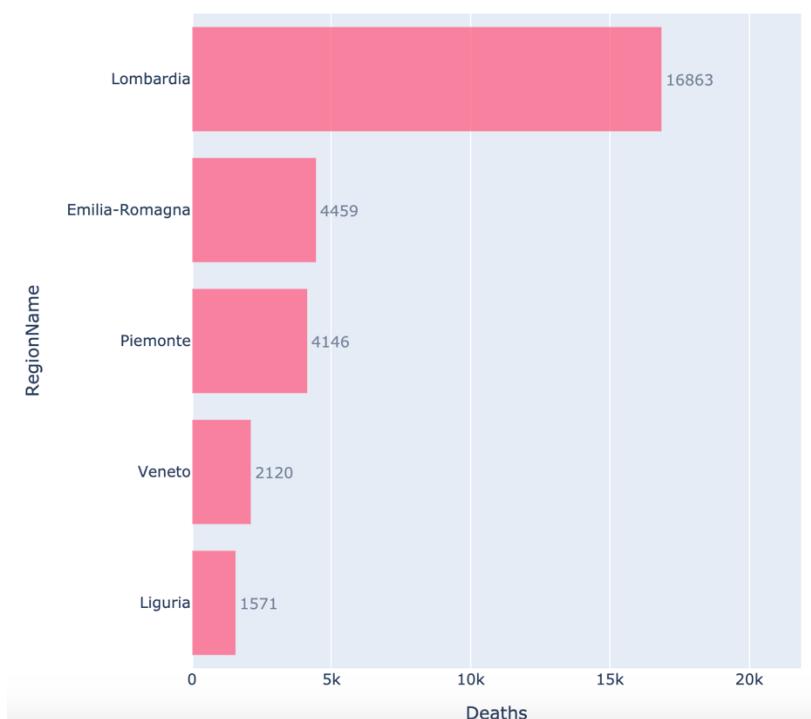
```
fig = px.bar(dgm.sort_values('TotalPositiveCases', ascending=False).head(5).sort_values('TotalPositiveCases', ascending=True),
             x="TotalPositiveCases", y="RegionName", title='Total Positive Cases', text='TotalPositiveCases', orientation='horizontal',
             width=700, height=700, range_x=[0, max(dgm['TotalPositiveCases'])+10000])
fig.update_traces(marker_color=tpc, opacity=0.6, textposition='outside')
fig.show()
```



- On observe que Lombardia est la région la plus touchée par Covid-19 avec 99940 cas positives.
- Aussi Piemonte par 32844 cas positives.
- Emilia-Romagna par 31805 cas positives.
- Veneto par 22864 cas positives.
- Toscana par 11785 cas positives.

## 2. Nombre de décès :

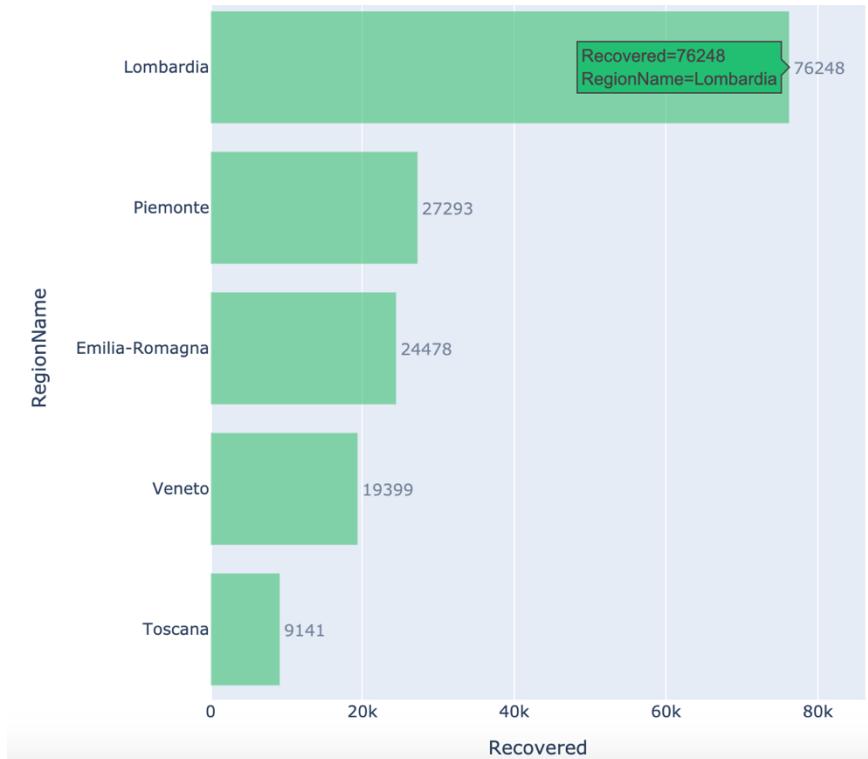
```
fig = px.bar(dgm.sort_values('Deaths', ascending=False).head(5).sort_values('Deaths', ascending=True),
             x="Deaths", y="RegionName", title='Deaths', text='Deaths', orientation='h',
             width=700, height=700, range_x=[0, max(dgm['Deaths'])+5000])
fig.update_traces(marker_color=dth, opacity=0.6, textposition='outside')
fig.show()
```



- On observe que Lombardia est la régions la plus touchées de Covid-19 par 16863 des décès.
- Aussi Emilia-Romagna par 4459 des décès
- Piemonte par 4146 de total des décès
- Veneto par 2120 des décès
- Liguria par 1571 des décès

## 3. Nombres de Guérisons :

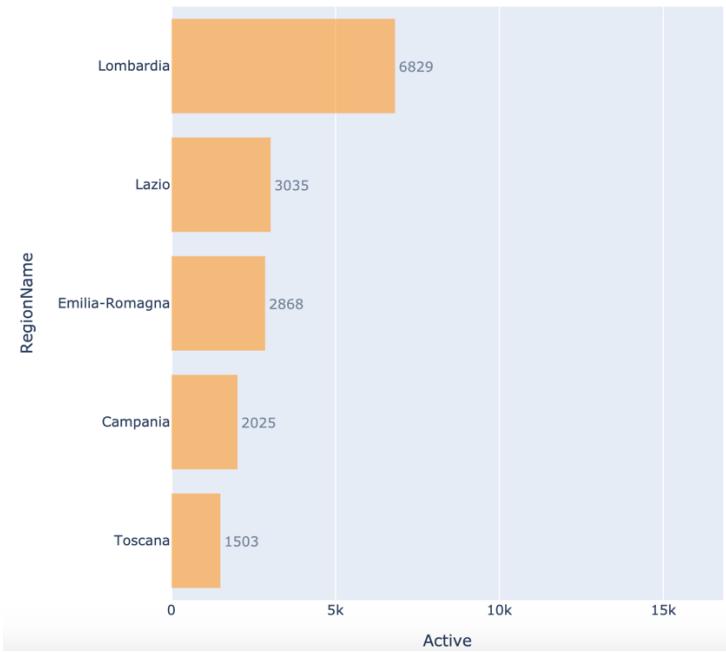
```
fig = px.bar(dgm.sort_values('Recovered', ascending=False).head(5).sort_values('Recovered', ascending=True),
              x="Recovered", y="RegionName", title='Recovered', text='Recovered', orientation='h',
              width=700, height=700, range_x=[0, max(dgm['Recovered'])+10000])
fig.update_traces(marker_color=rec, opacity=0.6, textposition='outside')
fig.show()
```



- On observe que Lombardia est la région qu'a le plus grand nombre de guérisons avec 76248 cas guérisons.
- Aussi Piemonte par 27293 de guérisons.
- Emilia-Romagna par 24478 de guérisons.
- Veneto par 19399 de guérisons.
- Toscana par 9141 de guérisons.

#### 4- Nombre des cas actives :

```
fig = px.bar(dgm.sort_values('Active', ascending=False).head(5).sort_values('Active', ascending=True),
             x="Active", y="RegionName", title='Currently Active', text='Active', orientation='h',
             width=700, height=700, range_x = [0, max(dgm['Active'])+10000])
fig.update_traces(marker_color=act, opacity=0.6, textposition='outside')
fig.show()
```



- On observe que Lombardia est la régions la plus touchées de Covid-19 par 6829 des cas active.
- Aussi Lazio par 3035 cas actives.
- Emilia-Romagna par 2868 cas actives.
- Campania par 2025 cas actives.
- Toscana par 1503 cas actives.

#### 4. Le taux de Mortalité :

```
# (Only regions with more than 500 case are considered)

dgm['Mortality Rate'] = round((dgm['Deaths']/dgm['TotalPositiveCases'])*100, 2)
temp = dgm[dgm['TotalPositiveCases']>500]
temp = temp.sort_values('Mortality Rate', ascending=False)

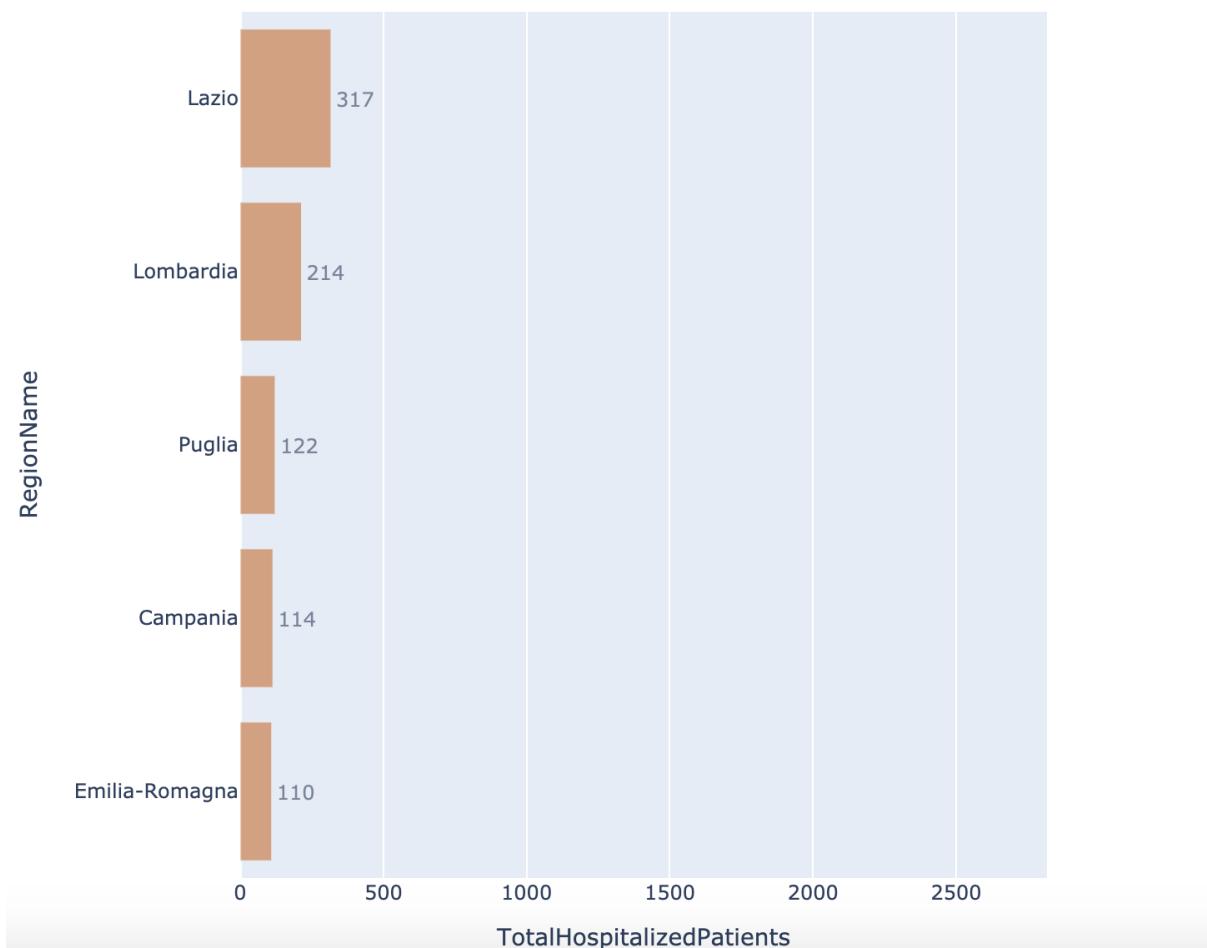
fig = px.bar(temp.sort_values('Mortality Rate', ascending=False).head(5).sort_values('Mortality Rate', ascending=True,
    x="Mortality Rate", y="RegionName", text="Mortality Rate", orientation='h',
    width=700, height=600, range_x = [0, 20], title='Mortality Rate (No. of Deaths Per 100 Confirmed Case)')
fig.update_traces(marker_color=dth, opacity=0.6, textposition='outside')
fig.show()
```



- On note que Lombardia a un taux de mortalité plus élevé que les autres régions avec un pourcentage de 16.87% ceci à cause de l'augmentation du nombre des décès dans la région Lombardia
- On note aussi que Liguria a un taux plus élevé avec un pourcentage 14.4 %
- Puis on a la région Emilia-Romagna avec un pourcentage de 14.02%
- Ainsi La région Marche avec un pourcentage de 13.64%
- Et dernièrement la région Piemonte avec un pourcentage de 12.62%

## 5. Nombre Total de patients hospitalisés :

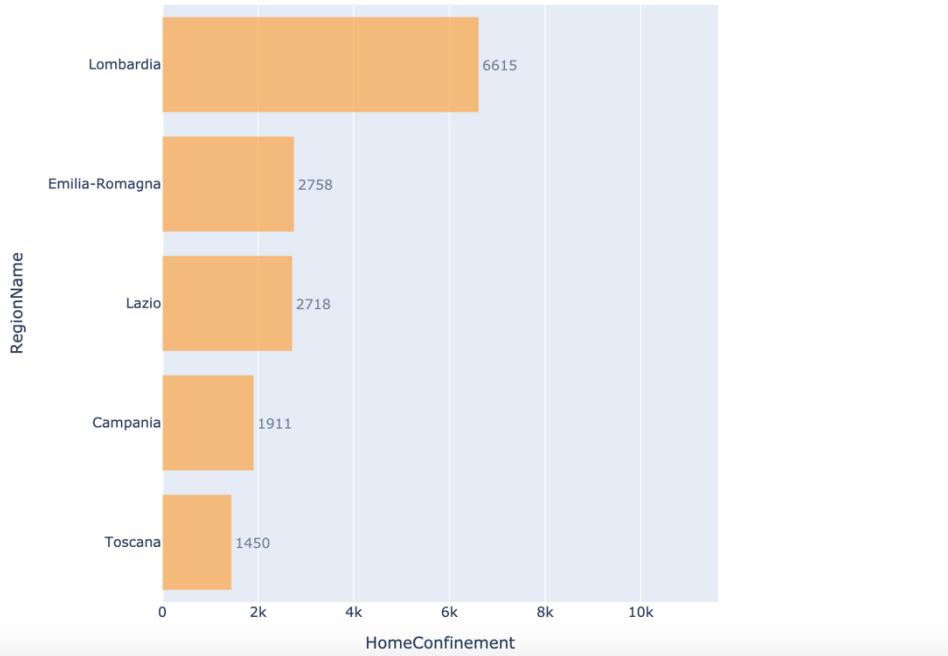
```
fig = px.bar(dgm.sort_values('TotalHospitalizedPatients', ascending=False).head(5).sort_values('TotalHospitalizedPatients', ascending=True), x="TotalHospitalizedPatients", y="RegionName", title='TotalHospitalizedPatients', text='TotalHospitalizedPatients', width=700, height=700, range_x=[0, max(dgm['TotalHospitalizedPatients'])+2500])
fig.update_traces(marker_color=hos, opacity=0.6, textposition='outside')
fig.show()
```



- On remarque que Lazio est la région la plus touchée par le Covid-19 avec 317 patients hospitalisés.
- Aussi Lombardia par 214 de nombre total des patients hospitalisés.
- Puglia par 122 de total de nombre total des patients hospitalisés.
- Campania par 114 de nombre total des patients hospitalisés.
- Emilia-Romagna par 110 de nombres total des patients hospitalisés.

## 6. Par Nombre de confinement à domicile :

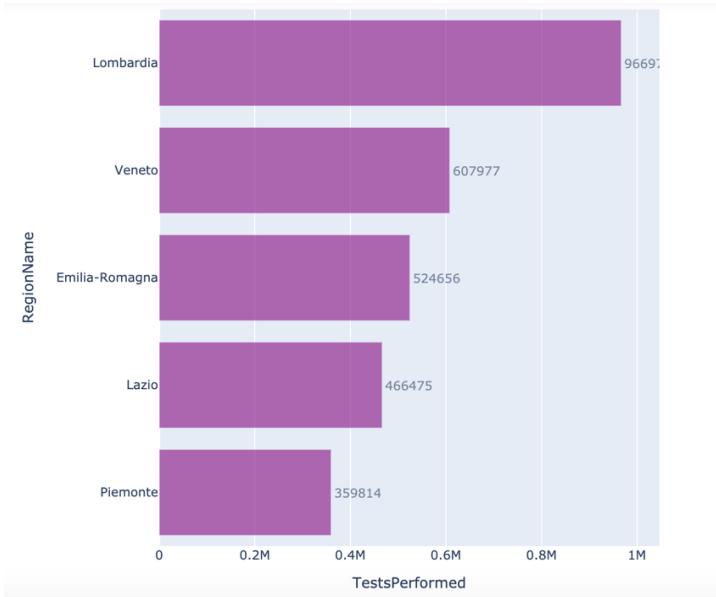
```
fig = px.bar(dgm.sort_values('HomeConfinement', ascending=False).head(5).sort_values('HomeConfinement', ascending=True,
x="HomeConfinement", y="RegionName", title='Home Confinement', text='HomeConfinement', orientation='h',
width=700, height=700, range_x=[0, max(dgm['HomeConfinement'])+5000])
fig.update_traces(marker_color=act, opacity=0.6, textposition='outside')
fig.show()
```



- Lombardia occupe la première place en ce qui concerne le nombre des patients en confinement à domicile.
- Aussi Emilia-Romagna : 2758 patients en confinement à domicile.
- Lazio : 2718 patients en confinement à domicile.
- Campania : 1911 patients en confinement à domicile.
- Toscana : 1450 patients en confinement à domicile.

## 7. Par Nombre des Tests Effectués :

```
fig = px.bar(dgm.sort_values('TestsPerformed', ascending=False).head(5).sort_values('TestsPerformed', ascending=True,
x="TestsPerformed", y="RegionName", title='Tests Performed (tests)', text='TestsPerformed', orientation='h',
width=700, height=700, range_x=[0, max(dgm['TestsPerformed'])+80000])
fig.update_traces(marker_color='purple', opacity=0.6, textposition='outside')
fig.show()
```



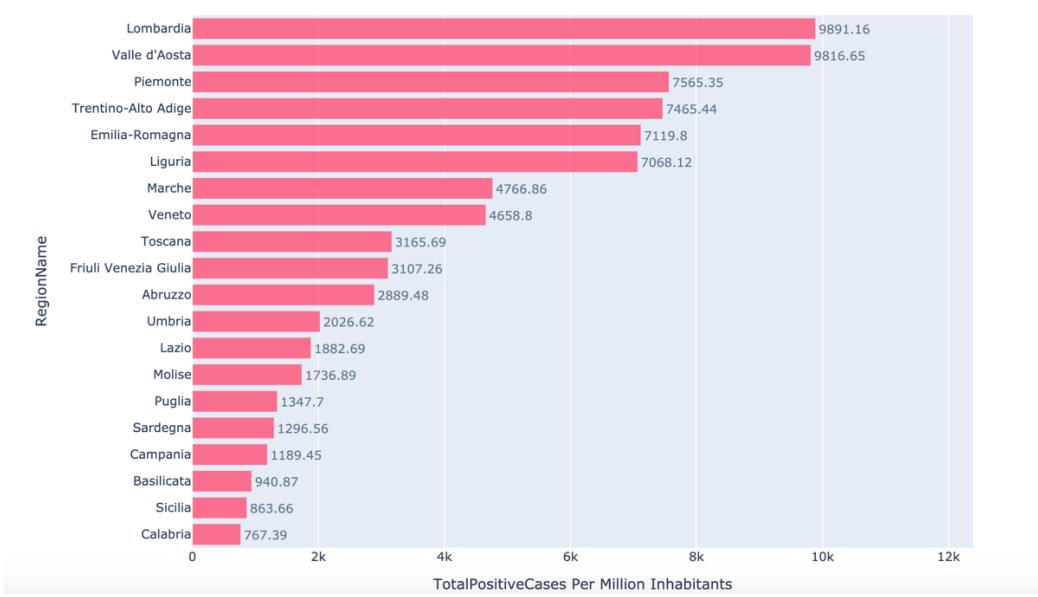
- On observe que la région Lombardia a effectué 966973 tests
- Veneto a effectué 607977 tests
- Emilia-Romagna a effectué 524656 tests
- Puis Lazio a effectué 466475 tests
- Dernièrement Piemonte a effectué 359814 tests

### Visualisation de total des cas positifs par million d'habitants :

```
# merge dataframes
temp = pd.merge(dgm_2, pop_reg, how='left', right_on='RegionName', left_on='RegionName')
# print(temp['Country Name'].isna())
temp = temp[['RegionName', 'TotalPositiveCases', 'Deaths', 'Recovered', 'Active', 'Population']]
#temp.columns = ['Region', 'TotalPositiveCases', 'Deaths', 'Recovered', 'Active', 'Population']

# calculate TotalPositiveCases/Population
temp['TotalPositiveCases Per Million Inhabitants'] = round(temp['TotalPositiveCases']/temp['Population']*1000000, 2)

fig = px.bar(temp.head(20).sort_values('TotalPositiveCases Per Million Inhabitants', ascending=True),
             x='TotalPositiveCases Per Million Inhabitants', y='RegionName', orientation='h',
             width=1000, height=700, text='TotalPositiveCases Per Million Inhabitants', title='Total Positive Cases per Million Inhabitants')
range_x = [0, max(temp['TotalPositiveCases Per Million Inhabitants'])+2500]
fig.update_traces(textposition='outside', marker_color='red', opacity=0.7)
fig.update_layout(uniformtext_minsize=8, uniformtext_mode='hide')
fig.show()
```

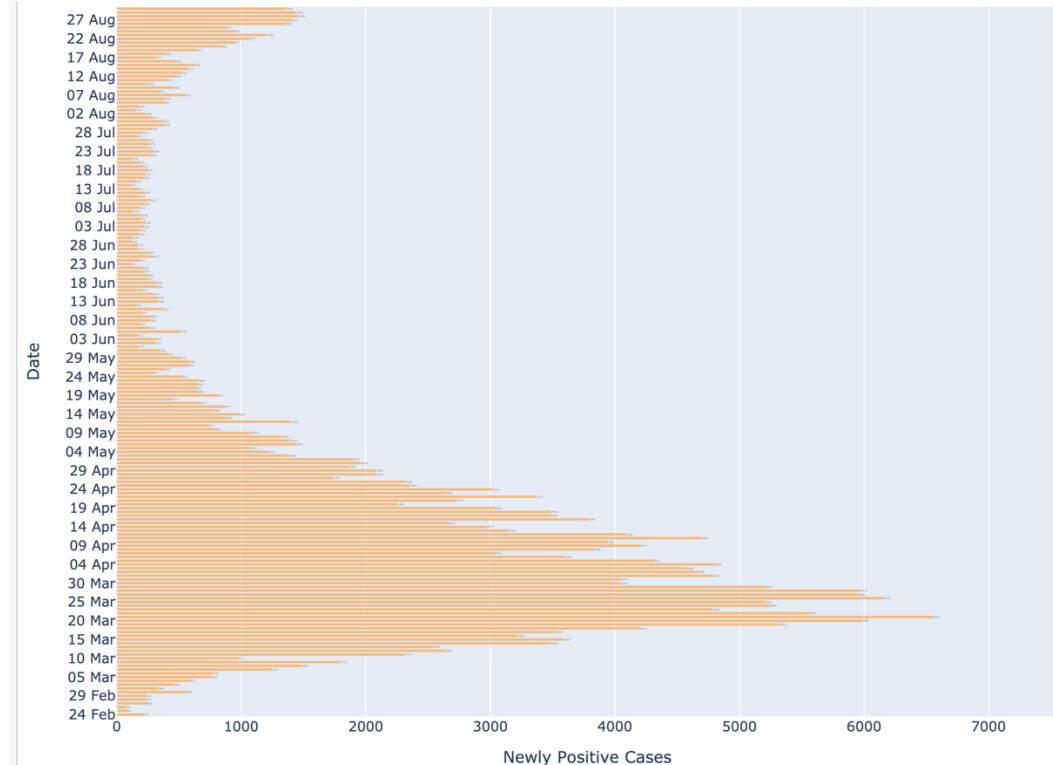


- On observe que Lombardia occupe la rang le plus élevés pour la visualisation de total des cas positive par million d'habitants avec une valeur 9891.16 cas positive par million d'habitants suivie par Valle d'Aosta avec une valeur 9816.65 cas positive par million d'habitants par contre Calabria occupe les rangs le plus bas avec une valeur 767.39 cas positive par million d'habitants.

## Visualisation Jour Par Jour :

```
temp = Data_Byregion.groupby('Date')['NewPositiveCases'].sum().reset_index()
temp['Date'] = pd.to_datetime(temp['Date'])
temp['Date'] = temp['Date'].dt.strftime('%d %b')

fig = px.bar(temp, x="NewPositiveCases", y="Date", orientation='h', height=800,
             text = 'NewPositiveCases',
             title='N. of New Positive Cases in Italy for each day',
             range_x = [0, max(temp['NewPositiveCases'])+1000])
fig.update_layout(xaxis_title='Newly Positive Cases')
fig.update_traces(marker_color=act, opacity=0.6, textposition='outside')
fig.show()
```



- Entre 24 Feb et 20 Mar : On note La croissance de nouveaux cas positives
- Par contre entre 20 Mar et 12 Aug : On note la décroissance de nouveaux cas positives
- Puis entre 12 Aug et 27 Aug : On note la croissance de nouveaux cas positives

## La Carte Graphiques :

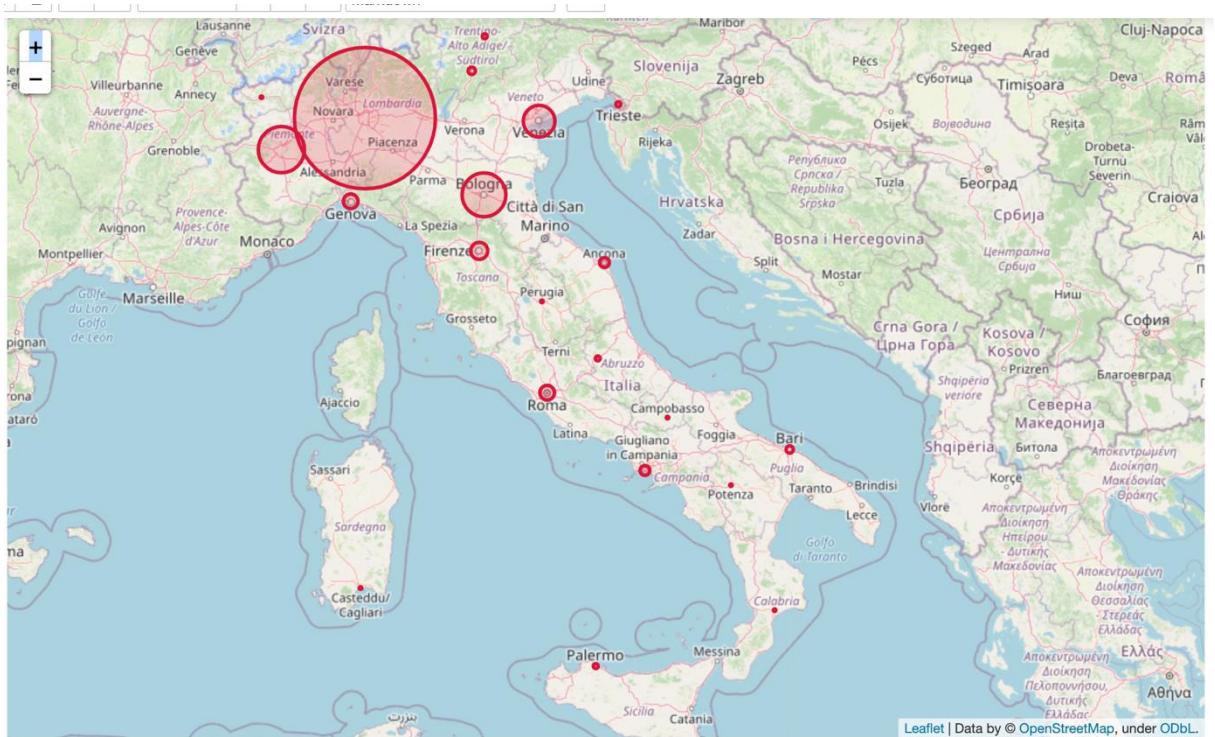
```
# Italy_Regions

m_Regions = folium.Map(location=[41.8719, 12.5674],
                      min_zoom=5, max_zoom=10, zoom_start=5)

for i in range(0, len(data)):
    folium.Circle(
        location=[data.iloc[i]['Latitude'], data.iloc[i]['Longitude']],
        color='crimson',
        fill = True,
        fill_color='crimson',
        tooltip = "<div style='margin: 0; background-color: black; color: white;'>"+
                  '<li><b>Country : '+str(data.iloc[i]['Country'])+'</b>' +
                  '<li><b>RegionName : '+str(data.iloc[i]['RegionName'])+'</b>' +
                  '<li><b>TotalPositiveCases : '+str(data.iloc[i]['TotalPositiveCases'])+'</b>' +
                  '<li><b>Deaths : '+str(data.iloc[i]['Deaths'])+'</b>' +
                  '<li><b>Recovered : '+str(data.iloc[i]['Recovered'])+'</b>' +
                  '<li><b>Active : '+str(data.iloc[i]['Active'])+'</b>' +
                  '<li>Taux de mortalité: "+ str(round((data.iloc[i]['Deaths']/data.iloc[i]['TotalPositiveCases'])))*
                  "</ul></div>",
        radius=int(data.iloc[i]['TotalPositiveCases'])*1).add_to(m_Regions)

m_Regions.save('m_Regions.html')

m_Regions
```



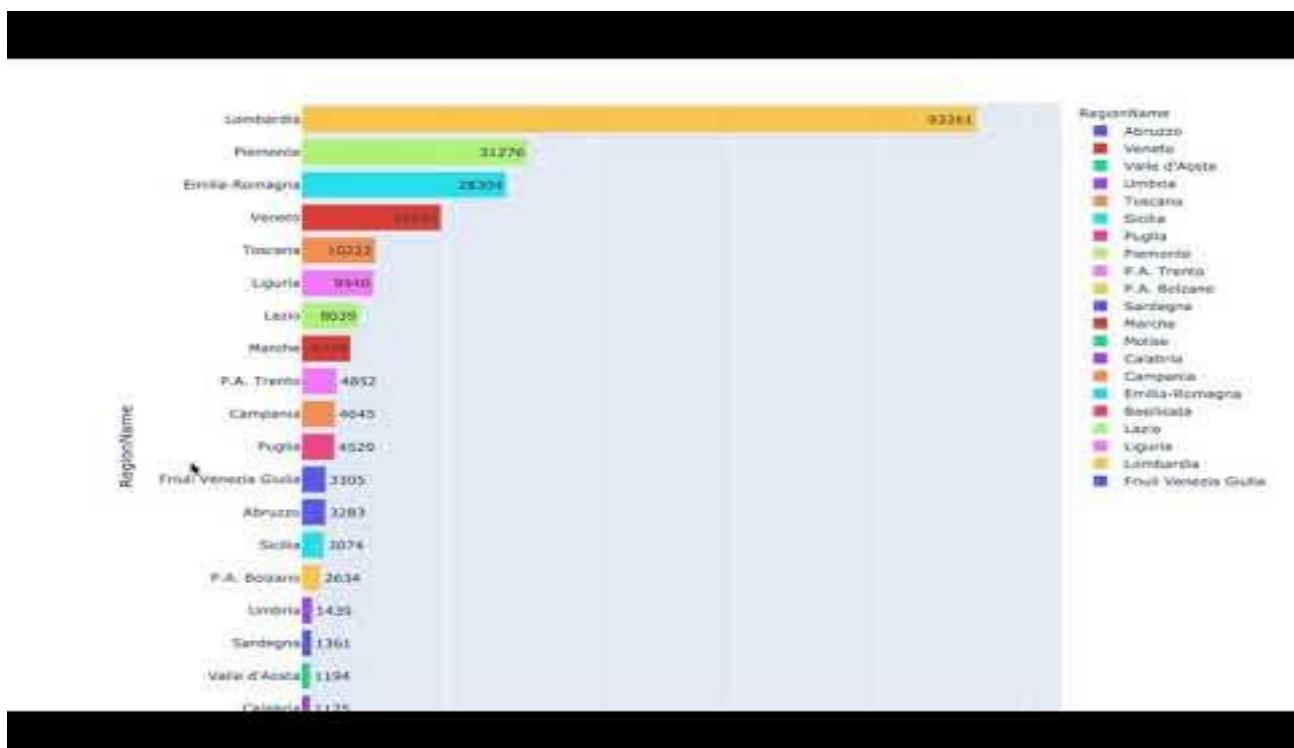
## Vidéo Visualisation De la carte graphique interactif par date :

Cette Vidéo montre le total des cas positive et le décès répartis par le temps ou la date dans une carte graphique.

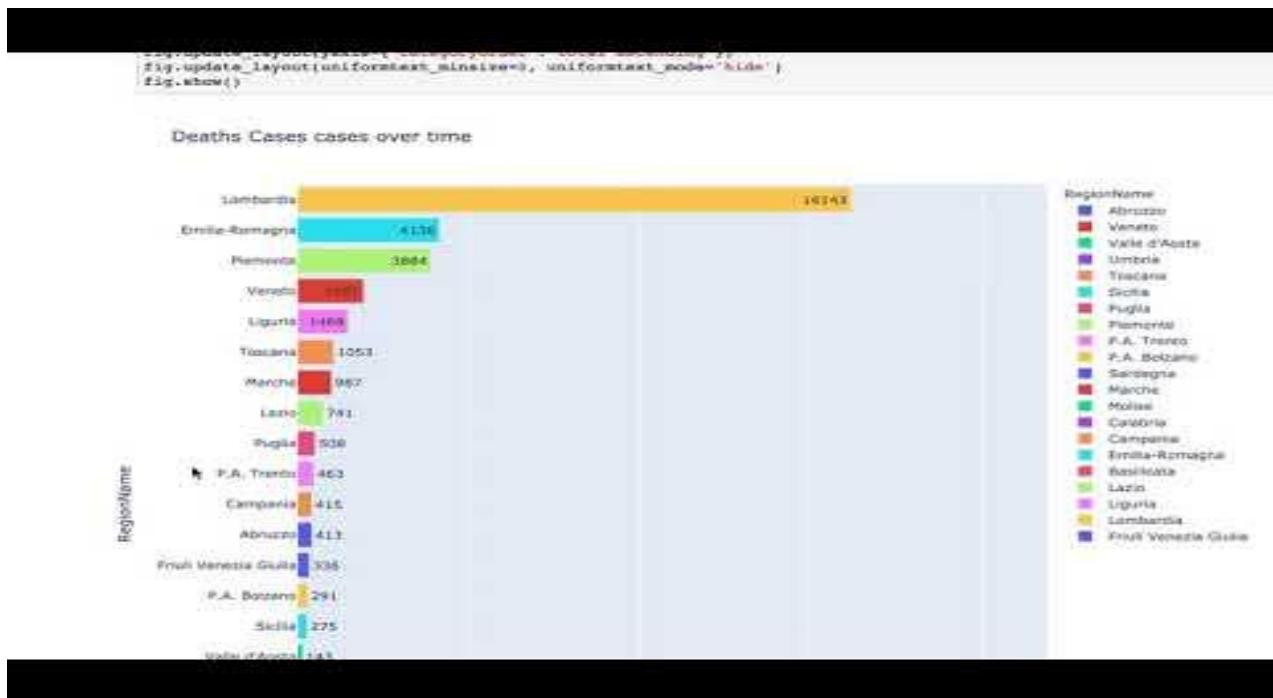


**Les vidéos de la visualisation Des total des cas positifs, les guérisons, les décès et les cas actives par répartitions de temps :**

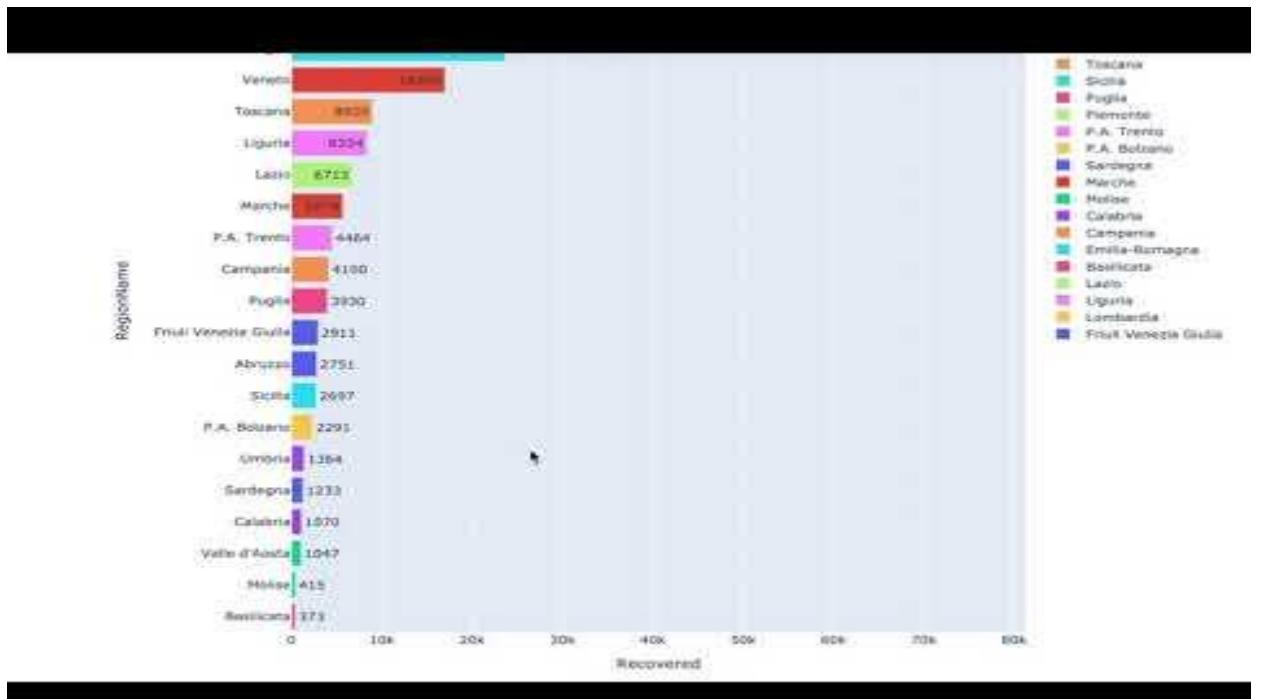
**1) Nombre total de cas positifs au fil du temps :**



## 2) Cas de décès Cas au fil du temps :



## 3) Cas récupérés au fil du temps :



## 4) Cas actifs au fil du temps :

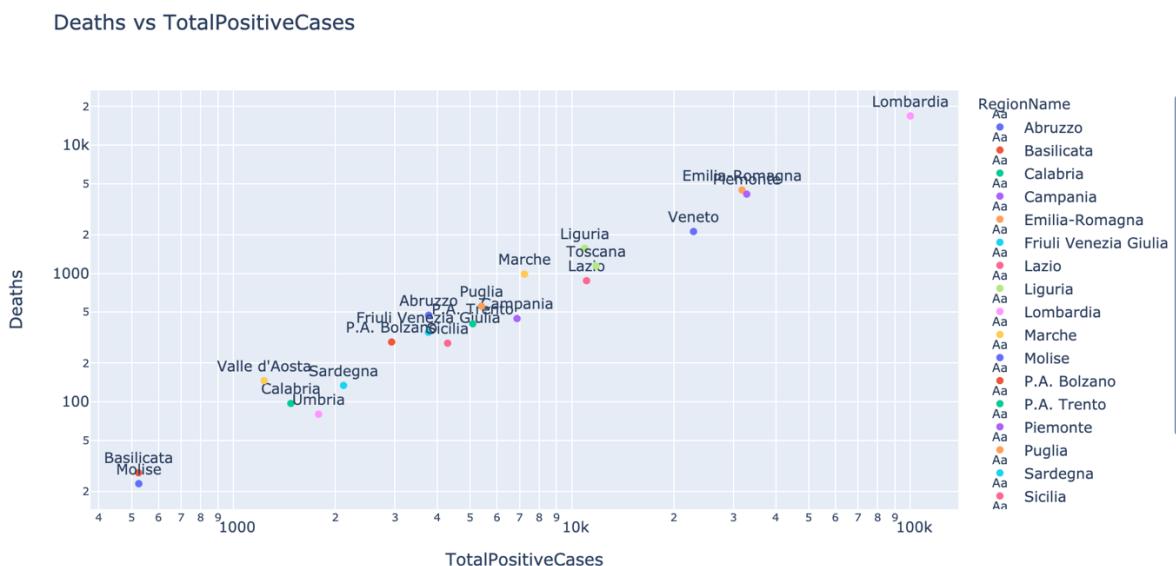


## Les cas des décès VS Le total des cas positives :

```

temp = data_grouped
fig = px.scatter(temp,
                 x='TotalPositiveCases', y='Deaths', color='RegionName',
                 text='RegionName', log_x=True, log_y=True, title='Deaths vs TotalPositiveCases')
fig.update_traces(textposition='top center')
fig.show()

```



Ce Graphe Montre le total des cas positifs par rapport à les cas de décès

### Durées de l'épidémie :

#### Remarque :

Dans le graphique, le dernier jour est indiqué comme un jour après la dernière notification d'un nouveau cas confirmé.

Gantt Chart



- On note d'après le graphe au-dessus que la durée de l'épidémie commence dans les dernières jours de mois février mais pour la fin de la durées de l'épidémie dans notre cas (par rapport à nos données), la fin de mois août et le début de mois septembre .

### Carte du calendrier :

```
Data_Byregion['Date'] = pd.to_datetime(Data_Byregion['Date'])
```

- Nombre de nouveaux cas confirmés chaque jour :

```
temp = Data_Byregion.groupby('Date')['TotalPositiveCases'].sum()
temp = temp.diff()

plt.figure(figsize=(20, 5))
ax = calmap.yearplot(temp, fillcolor='white', cmap='Oranges', linewidth=0.5)
```



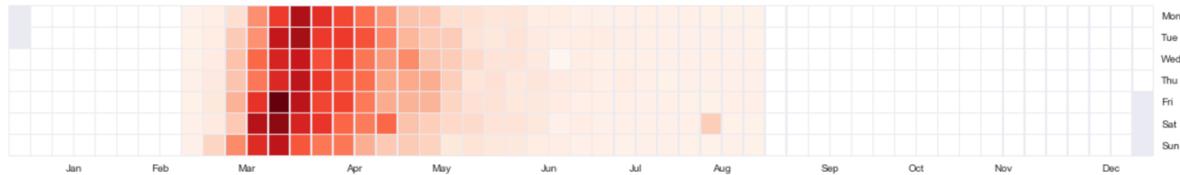
- Nombre de nouveaux décès chaque jour :

```

: temp = Data_Byregion.groupby('Date')['Deaths'].sum()
temp = temp.diff()

plt.figure(figsize=(20, 5))
ax = calmap.yearplot(temp, fillcolor='white', cmap='Reds', linewidth=0.5)

```



- Nombre de régions nouvellement touchées chaque jour :

```

: spread = Data_Byregion[Data_Byregion['TotalPositiveCases']!=0].groupby('Date')
spread = spread['RegionName'].unique().apply(len).diff()

plt.figure(figsize=(20, 5))
ax = calmap.yearplot(spread, fillcolor='white', cmap='Greens', linewidth=0.5)

```

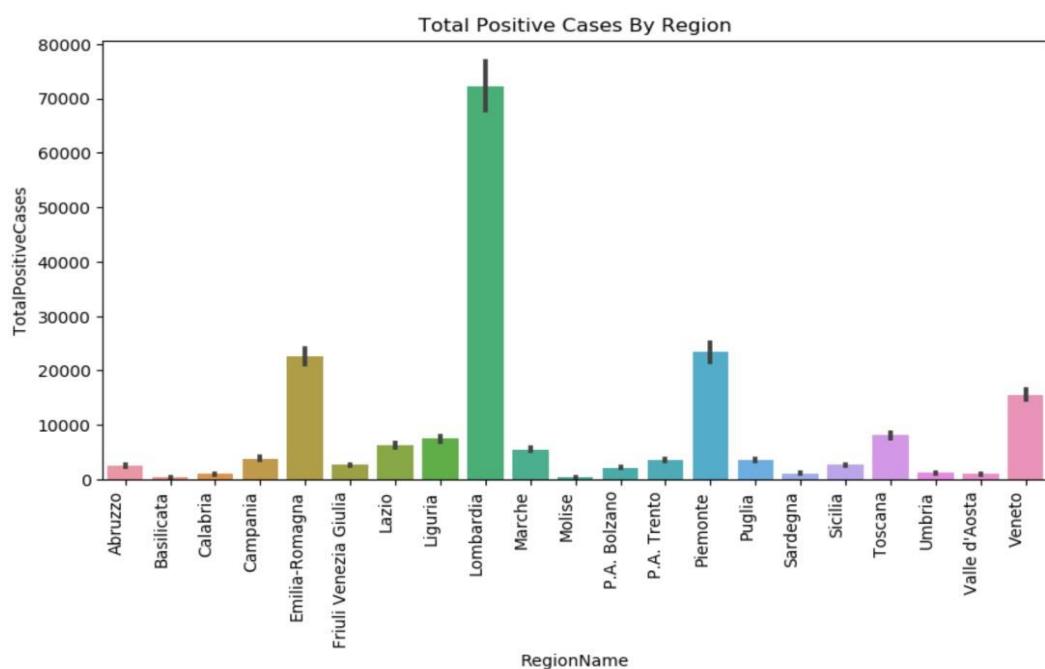


### Total des cas positifs par région :

```

sns.barplot(x=Data_Byregion['RegionName'], y=Data_Byregion['TotalPositiveCases'])
plt.xticks(rotation=90, ha='right')
plt.title('Total Positive Cases By Region')
plt.rc('figure', figsize=(10, 5))

```

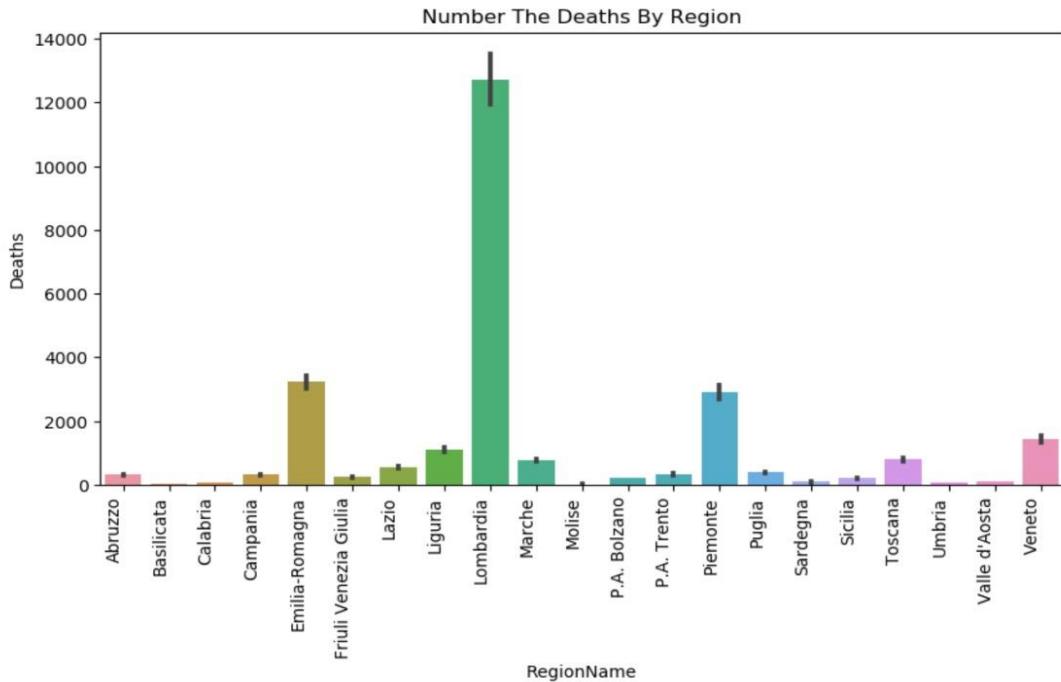


- Dans ce graphe on remarque que la région de Lombardia est plus touchée par l'épidémie dépassant 70000 cas positifs suivis par la région Emilia-Romagna et la région Piemonte

dépassent 20000 cas positive, suivis par la région Veneto dépassant plus que 10500 cas positive et le reste des régions sont inférieurs à 10000 cas positive.

### Les Décès par région :

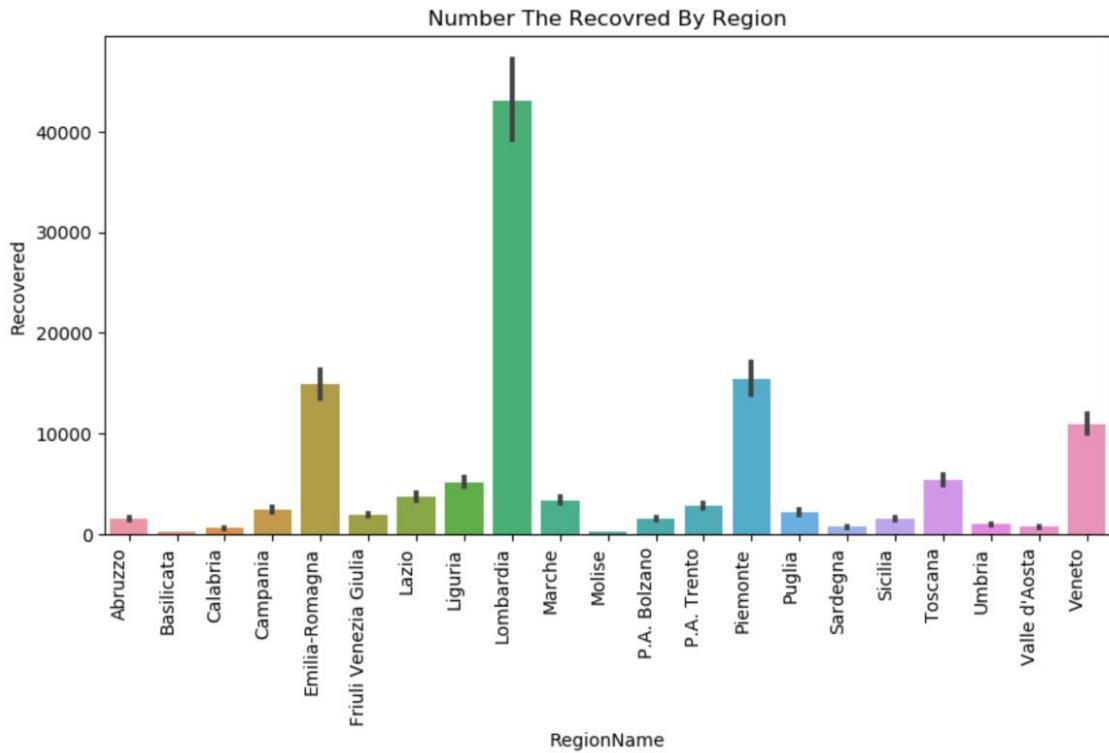
```
sns.barplot(x=Data_Byregion['RegionName'], y=Data_Byregion['Deaths'])
plt.xticks(rotation=90, ha='right')
plt.title('Number The Deaths By Region')
plt.rc('figure', figsize=(10, 5))
```



- Dans ce graphe on remarque que la région de Lombardia est plus touchée par l'épidémie dépassant 12000 cas de décès suivis par la région Emilia-Romagna de plus de 2600 décès, suivis par la région Piemonte dépassant plus que 2500 de décès et le reste des régions sont inférieurs à 1800 cas de décès.

### Les guérisons par région :

```
sns.barplot(x=Data_Byregion['RegionName'], y=Data_Byregion['Recovered'])
plt.xticks(rotation=90, ha='right')
plt.title('Number The Recovered By Region')
plt.rc('figure', figsize=(10, 5))
```



- Dans ce graphe on constate que la région de Lombardia est plus touchée par l'épidémie dépassant 40000 guérisons suivis par la région Emilia-Romagna et la région Piemonte et Veneto dépassent 10000 guérisons et le reste des régions sont inférieurs à 10000 guérisons.

### Les trois régions les plus touchés (Boite à Moustache) :

*Pour le total des cas positifs :*

```
plt.figure(figsize=(10,5), dpi=100)
plt.style.use('default')

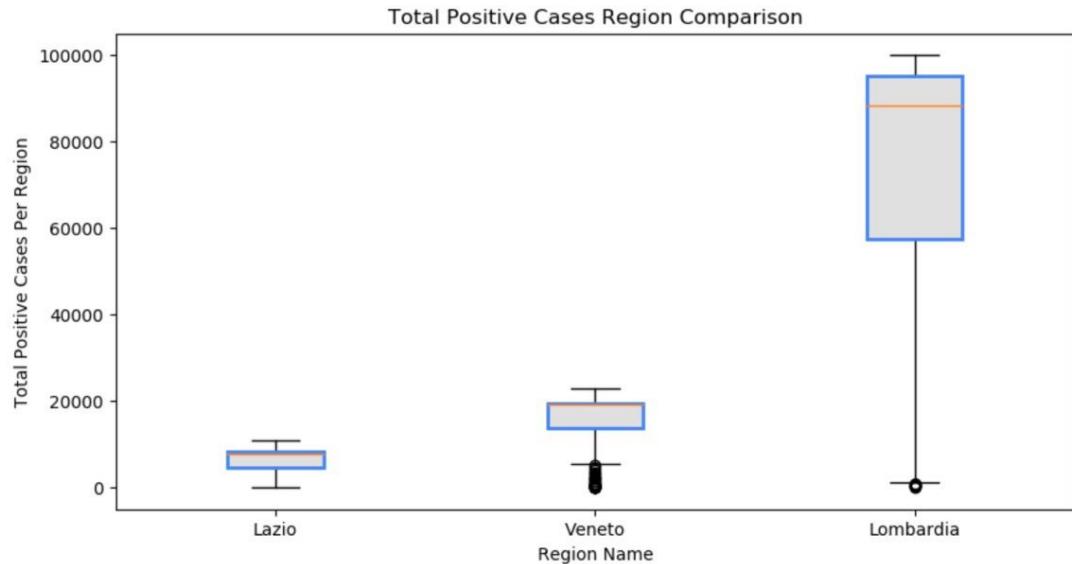
Lazio = Data_Byregion.loc[Data_Byregion['RegionName'] == "Lazio"]['TotalPositiveCases']
Veneto = Data_Byregion.loc[Data_Byregion['RegionName'] == "Veneto"]['TotalPositiveCases']
Lombardia = Data_Byregion.loc[Data_Byregion['RegionName'] == "Lombardia"]['TotalPositiveCases']

bp = plt.boxplot([Lazio, Veneto, Lombardia], labels=['Lazio', 'Veneto', "Lombardia"], patch_artist=True)

plt.title('Total Positive Cases Region Comparison')
plt.ylabel('Total Positive Cases Per Region')
plt.xlabel('Region Name')

for box in bp['boxes']:
    #Set edge color:
    box.set(color='#4286f4', linewidth=2)
    # Change Fill Color:
    box.set(facecolor = '#e0e0e0')

plt.show()
```



*Pour les guérisons :*

```

plt.figure(figsize=(10,5), dpi=100)
plt.style.use('default')

Lazio = Data_Byregion.loc[Data_Byregion['RegionName'] == "Lazio"]['Recovered']
Veneto = Data_Byregion.loc[Data_Byregion['RegionName'] == "Veneto"]['Recovered']
Lombardia = Data_Byregion.loc[Data_Byregion['RegionName'] == "Lombardia"]['Recovered']

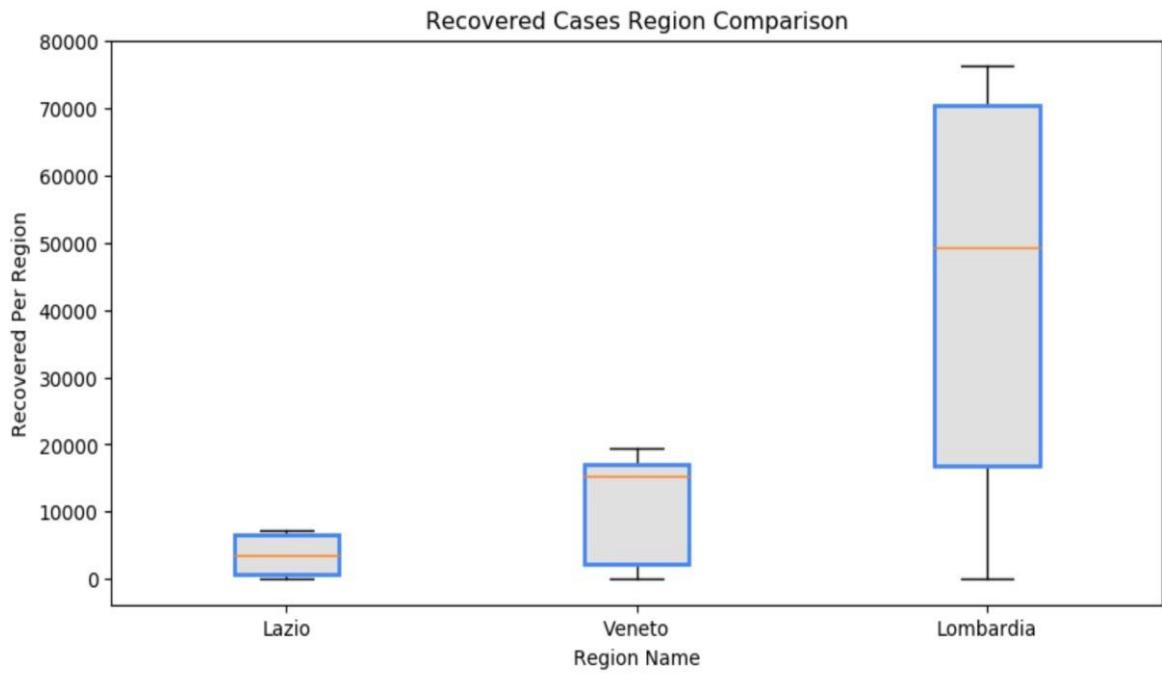
bp = plt.boxplot([Lazio, Veneto, Lombardia], labels=['Lazio', 'Veneto', "Lombardia"], patch_artist=True)

plt.title('Recovered Cases Region Comparison')
plt.ylabel('Recovered Per Region')
plt.xlabel('Region Name')

for box in bp['boxes']:
    #Set edge color:
    box.set(color="#4286f4", linewidth=2)
    # Change Fill Color:
    box.set(facecolor = '#e0e0e0')

plt.show()

```



*Pour les décès :*

```

plt.figure(figsize=(10,5), dpi=100)
plt.style.use('default')

Lazio = Data_Byregion.loc[Data_Byregion['RegionName'] == "Lazio"]['Deaths']
Veneto = Data_Byregion.loc[Data_Byregion['RegionName'] == "Veneto"]['Deaths']
Lombardia = Data_Byregion.loc[Data_Byregion['RegionName'] == "Lombardia"]['Deaths']

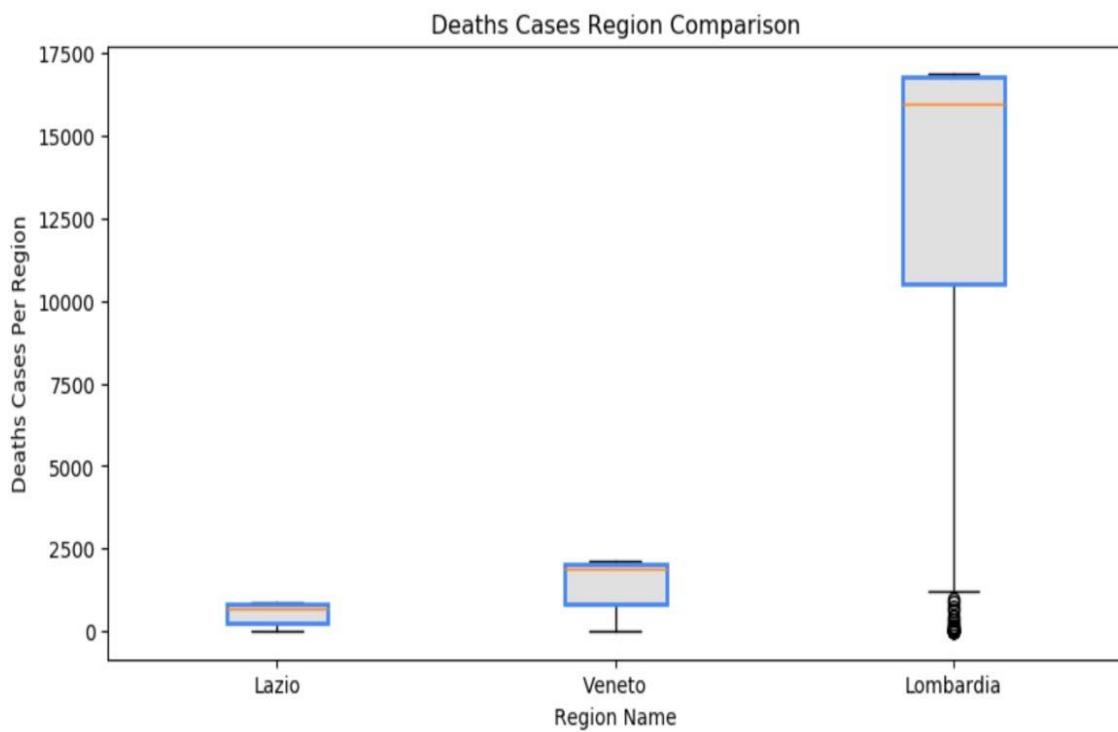
bp = plt.boxplot([Lazio, Veneto, Lombardia], labels=['Lazio', 'Veneto', "Lombardia"], patch_artist=True)

plt.title('Deaths Cases Region Comparison')
plt.ylabel('Deaths Cases Per Region')
plt.xlabel('Region Name')

for box in bp['boxes']:
    #Set edge color:
    box.set(color="#4286f4", linewidth=2)
    # Change Fill Color:
    box.set(facecolor = '#e0e0e0')

plt.show()

```



### Les Histogrammes par région :

*Pour le Total des cas positifs :*

```

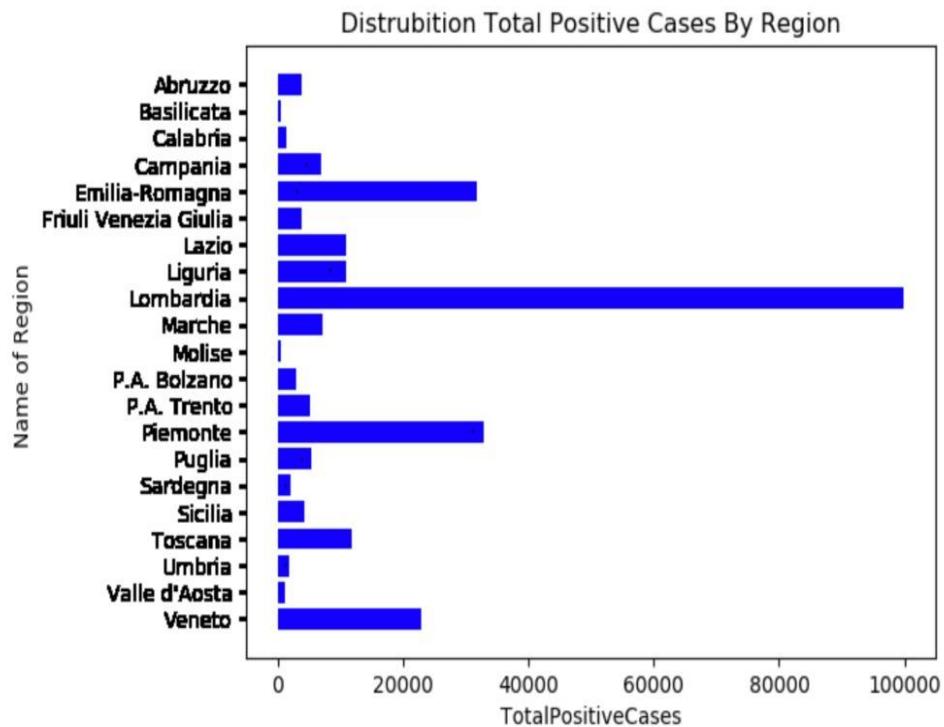
plt.rcdefaults()
fig, ax = plt.subplots()

# Importation Des Données
Region = Data_Byregion['RegionName']
error = np.random.rand(len(Region))
TotalPositiveCases = Data_Byregion['TotalPositiveCases']

ax.bart(Region, TotalPositiveCases, xerr=error, align='center', color='b')
ax.set_yticks(Region)
ax.set_yticklabels(Region)
ax.invert_yaxis() # labels read top-to-bottom
ax.set_xlabel('TotalPositiveCases')
ax.set_ylabel('Name of Region ')
ax.set_title('Distrubition Total Positive Cases By Region')

plt.show()

```



*Pour les Décès :*

```

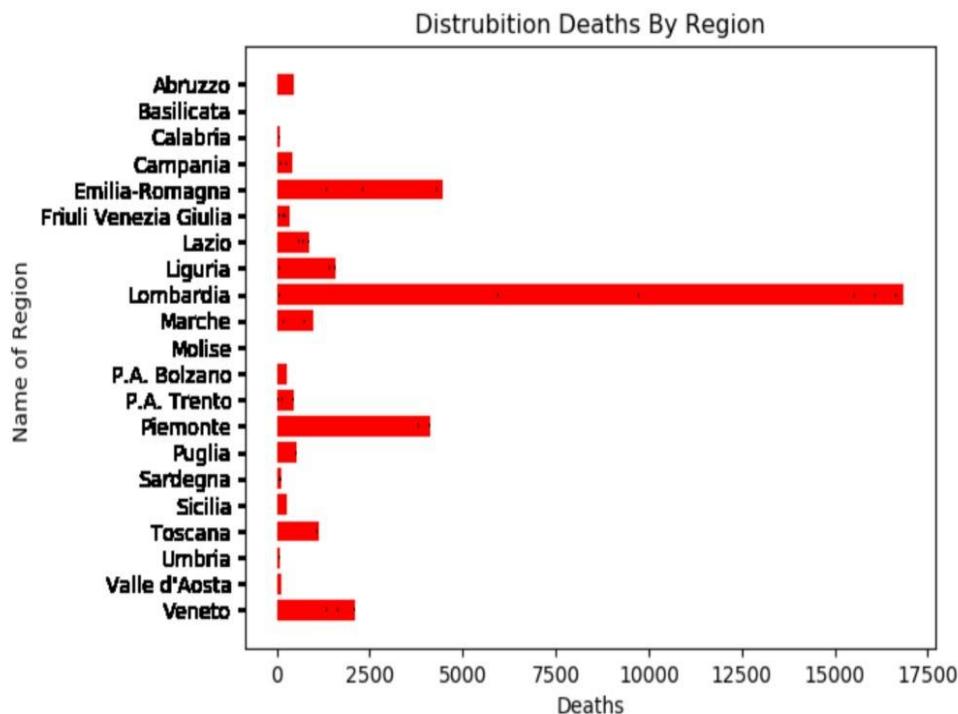
plt.rcParams()
fig, ax = plt.subplots()

# Importation Des Données
Region = Data_Byregion['RegionName']
error = np.random.rand(len(Region))
Deaths = Data_Byregion['Deaths']

ax.barh(Region, Deaths, xerr=error, align='center', color='Red')
ax.set_yticks(Region)
ax.set_yticklabels(Region)
ax.invert_yaxis() # labels read top-to-bottom
ax.set_xlabel('Deaths')
ax.set_ylabel('Name of Region ')
ax.set_title('Distribution Deaths By Region')

plt.show()

```



*Pour les guérisons :*

```

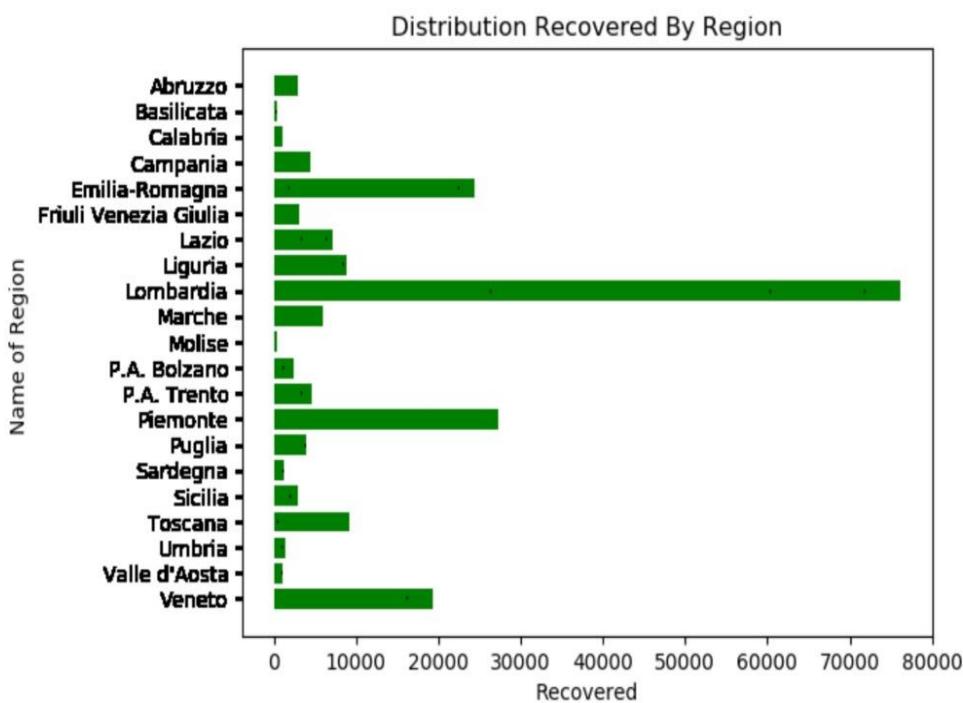
plt.rcParams()
fig, ax = plt.subplots()

# Importation Des Données
Region = Data_Byregion['RegionName']
error = np.random.rand(len(Region))
Recovered = Data_Byregion['Recovered']

ax.barh(Region, Recovered, xerr=error, align='center', color='Green')
ax.set_yticks(Region)
ax.set_yticklabels(Region)
ax.invert_yaxis() # labels read top-to-bottom
ax.set_xlabel('Recovered')
ax.set_ylabel('Name of Region ')
ax.set_title('Distribution Recovered By Region')

plt.show()

```



Pour les cas Actives :

```

np.random.seed(19680801)

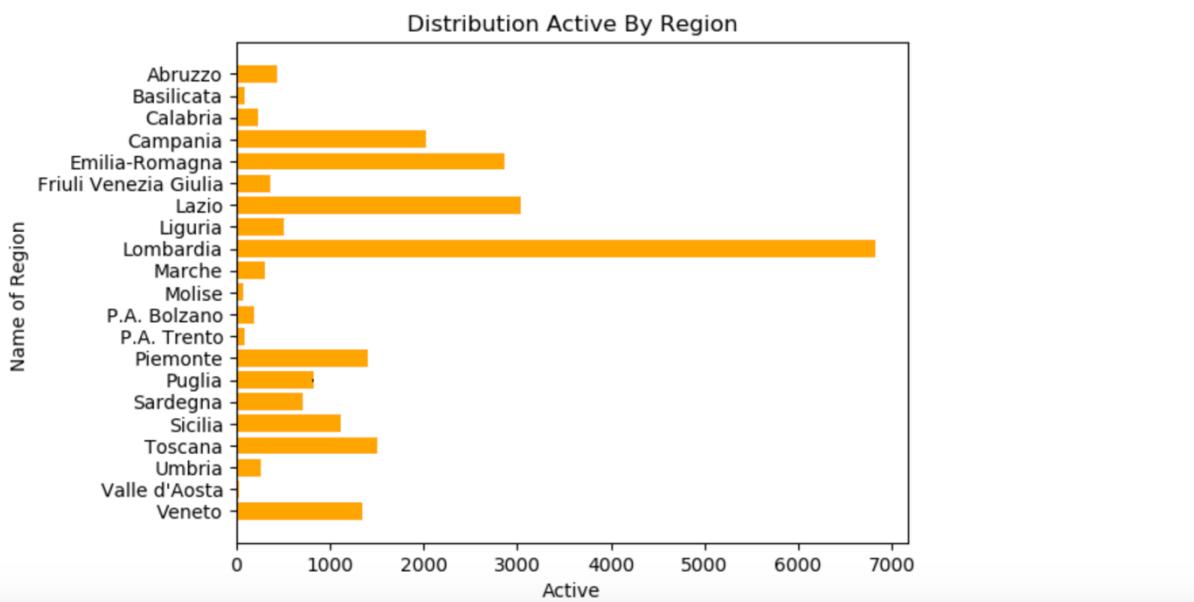
plt.rcParams()
fig, ax = plt.subplots()

# Importation Des Données
Region = data['RegionName']
error = np.random.rand(len(Region))
Active = data['Active']

ax.barh(Region, Active, xerr=error, align='center', color='Orange')
ax.set_yticks(Region)
ax.set_yticklabels(Region)
ax.invert_yaxis() # labels read top-to-bottom
ax.set_xlabel('Active')
ax.set_ylabel('Name of Region ')
ax.set_title('Distribution Active By Region')

plt.show()

```



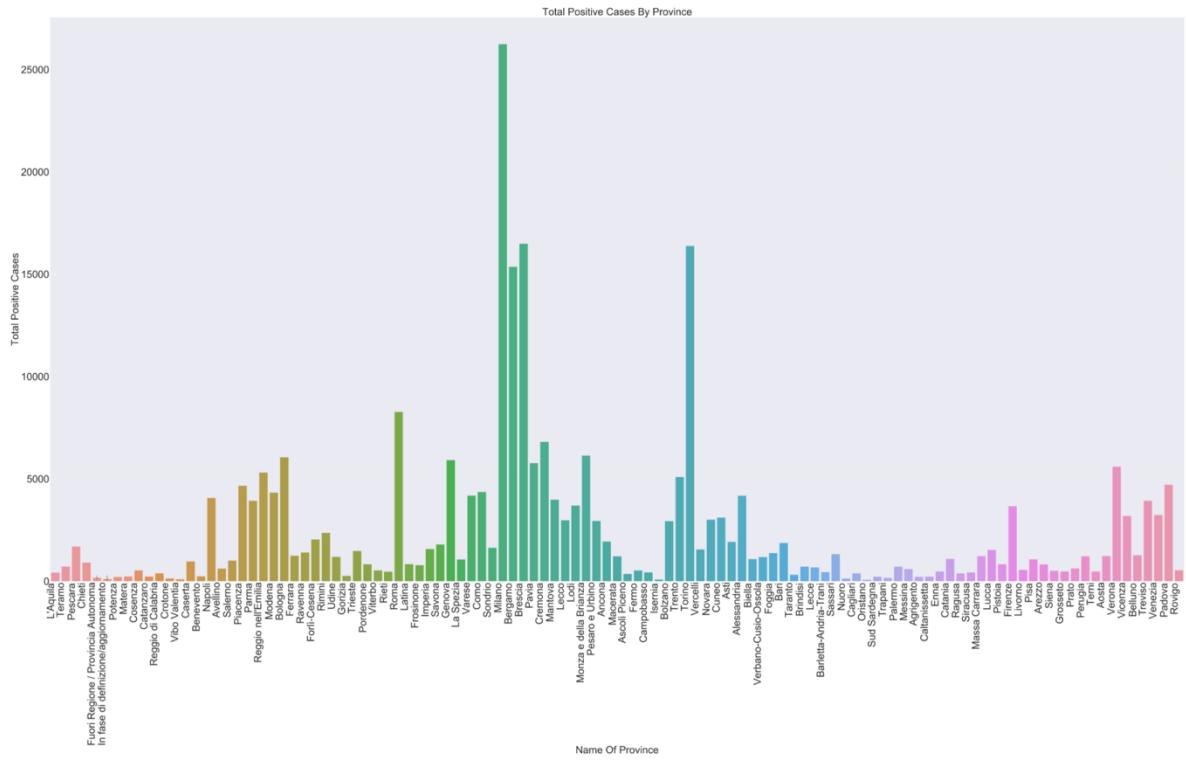
## Pour Table2 :(Par Province)

**Total des cas positifs par région :**

```
#Province Bar Plot
sns.barplot(x=Data_Byprovince['ProvinceName'], y=Data_Byprovince['TotalPositiveCases'])
plt.xticks(rotation=90, size=80, ha='right')
plt.yticks(size=80, ha='right')

plt.title('Total Positive Cases By Province',size=80)
plt.ylabel('Total Positive Cases', size=80)
xlabel=plt.xlabel('Name Of Province', size=80)

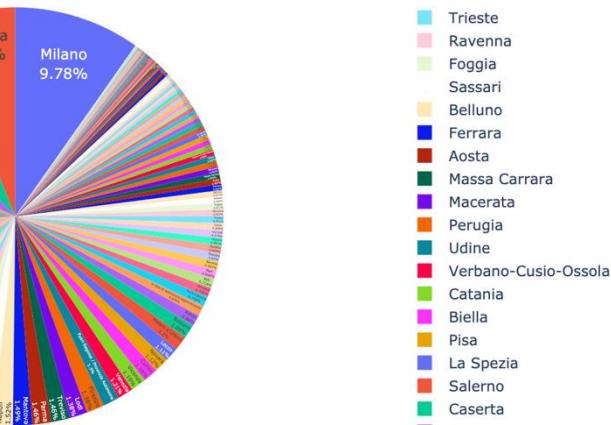
plt.rc('figure', figsize=(160,80))
```



## Visualisation Des Pies :

```
#Distribution of Total Positive Cases Per Province
fig = px.pie(data_grouped_province, values=data_grouped_province['TotalPositiveCases'], names=data_grouped_province['Province'],
              title='Distribution of Total Positive Cases Per Province',
              )
fig.update_traces(textposition='inside', textinfo='percent+label')
fig.update_layout(
    template='plotly_white')
fig.show()
```

## Distribution of Total Positive Cases Per Province



- Ce pie Graphe note que la province de Milano se classe la première avec pourcentage de 9.78%, suivi par la province Brescia avec un pourcentage 6.15%, suivi par Torino avec un pourcentage de 6.11% puis Bergamo avec un pourcentage de 5.73% et par rapport à les autres provinces le pourcentage est inférieur à 4%.

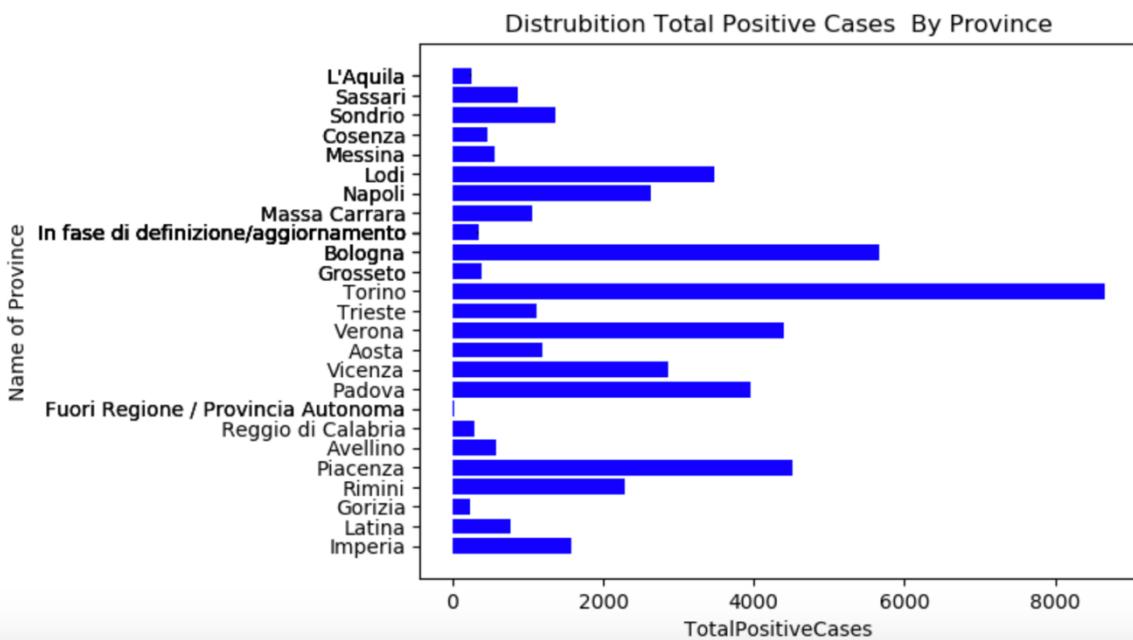
## Histogrammes De Total Positive Cases :

```
plt.rcParams()
fig, ax = plt.subplots()

# Importation Des Données
Province = Data_Byprovince['ProvinceName'][::600]
error = np.random.rand(len(Province))
TotalPositiveCases = Data_Byprovince['TotalPositiveCases'][::600]

ax.barh(Province, TotalPositiveCases, xerr=error, align='center', color='b')
ax.set_yticks(Province)
ax.set_yticklabels(Province)
plt.setp(ax.get_yticklabels(), rotation=360, ha="right",
         rotation_mode="anchor")
ax.invert_yaxis() # labels read top-to-bottom
ax.set_xlabel('TotalPositiveCases')
ax.set_ylabel('Name of Province')
ax.set_title('Distrubition Total Positive Cases By Province')

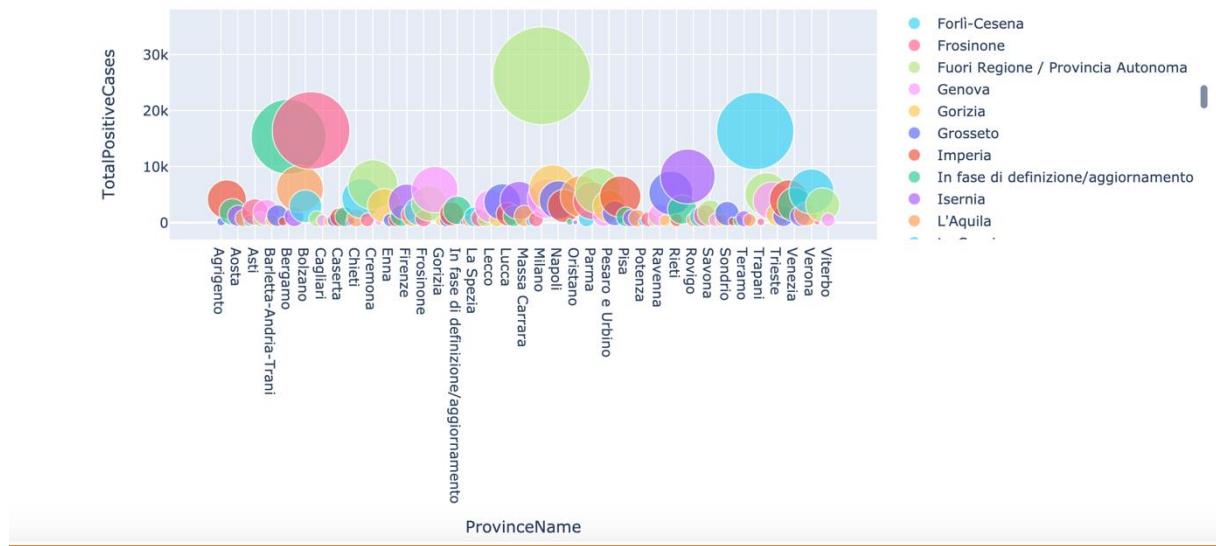
plt.show()
```



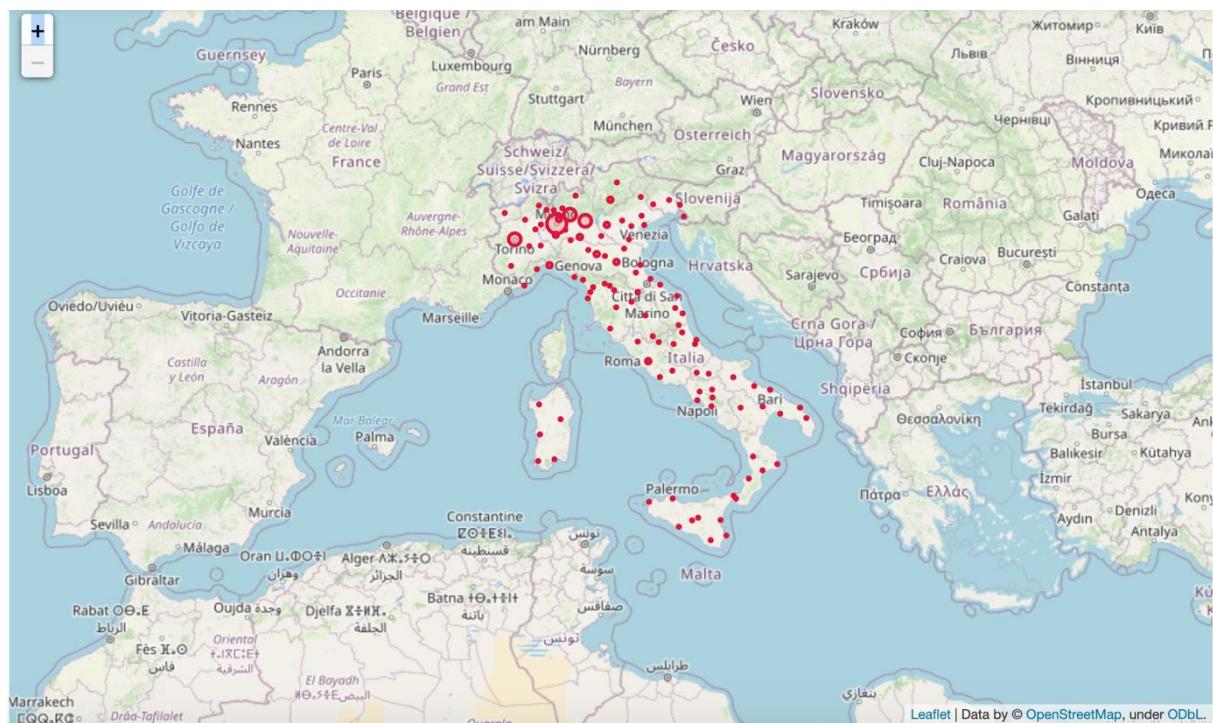
## Scatter Plot Par Province :

```
fig = px.scatter(data_grouped_province, x=data_grouped_province['ProvinceName'], y=data_grouped_province['TotalPositiveCases'],
                  color=data_grouped_province['ProvinceName'], hover_name=data_grouped_province['ProvinceName'], size_=

fig.update_layout()
fig.show()
```



### La Carte Graphique De Total des cas positive :



### Partie 3 : Étude Statistique

- Pour connaitre le nombre de cas à la date maximale l'erreur serait de faire la somme de la colonne TotalPositiveCases, car il s'agit d'une donnée cumulée chaque jour !
- Il faut donc extraire les données à la date souhaitée et de réaliser ensuite les différents calculs
- On extrait les données à la dernière date la plus récente

```
.2]: dataDerniereDate = Data_Byregion[Data_Byregion['Date'] == max(Data_Byregion['Date'])].reset_index()

.3]: TotalPositiveCases = dataDerniereDate["TotalPositiveCases"].sum()
TotalPositiveCases

268218

.4]: Recovered = dataDerniereDate["Recovered"].sum()
Recovered

208536

.5]: Deaths = dataDerniereDate["Deaths"].sum()
Deaths

35477
```

## Les Statistiques de toute l'Italie :

```
print(" Confirme : "+str(TotalPositiveCases))
print(" Gueris : "+str(Recovered))
print(" Decedes : "+str(Deaths))
print(" Taux mortalité (%): "+str(round((Deaths/TotalPositiveCases)*100,2)))

#Pour utilisation dans la synthèse
confirme = TotalPositiveCases
gueris = Recovered
decedes = Deaths
mortalite = round((Deaths/TotalPositiveCases)*100,2)

Confirme : 268218
Gueris : 208536
Decedes : 35477
Taux mortalité (%): 13.23
```

## Déterminer les caractéristiques de la variable des guérisons "Recovered":

```
print("la moyenne est : ",data["Recovered"].mean())
#la variance :
print("la variance est : ",data["Recovered"].var())
#l'écart type :
print("l'écart type est : ",data["Recovered"].std())
#le Min,Max et la somme :
print("le minimum de la série est : ",min(data["Recovered"]))
print("le maximum de la série est : ",max(data["Recovered"]))
print("la somme des éléments de la série est : ",sum(data["Recovered"]))
#la médiane :
print("la médiane est : ",data["Recovered"].median())

la moyenne est : 9930.285714285714
la variance est : 290316646.11428577
l'écart type est : 17038.68087952485
le minimum de la série est : 408
le maximum de la série est : 76248
la somme des éléments de la série est : 208536
la médiane est : 4029.0
```

## Déterminer les caractéristiques de la variable des décès "Deaths":

```

1: #Déterminer les caractéristiques de la variable des décès "Deaths":
#la moyenne :
print("la moyenne est : ",data["Deaths"].mean())
#la variance :
print("la variance est : ",data["Deaths"].var())
#l'écart type :
print("l'écart type est : ",data["Deaths"].std())
#le Min,Max et la somme :
print("le minimum de la série est : ",min(data["Deaths"]))
print("le maximum de la série est : ",max(data["Deaths"]))
print("la somme des éléments de la série est : ",sum(data["Deaths"]))
#la médiane :
print("la médiane est : ",data["Deaths"].median())

la moyenne est : 1689.3809523809523
la variance est : 13637325.047619045
l'écart type est : 3692.874902784962
le minimum de la série est : 23
le maximum de la série est : 16863
la somme des éléments de la série est : 35477
la médiane est : 445.0

```

## Déterminer les caractéristiques de la variable du Total des cas positifs "TotalPositiveCases":

```

print("la moyenne est : ",data["TotalPositiveCases"].mean())
#la variance :
print("la variance est : ",data["TotalPositiveCases"].var())
#l'écart type :
print("l'écart type est : ",data["TotalPositiveCases"].std())
#le Min,Max et la somme :
print("le minimum de la série est : ",min(data["TotalPositiveCases"]))
print("le maximum de la série est : ",max(data["TotalPositiveCases"]))
print("la somme des éléments de la série est : ",sum(data["TotalPositiveCases"]))
#la médiane :
print("la médiane est : ",np.median(data["TotalPositiveCases"]))

la moyenne est : 12772.285714285714
la variance est : 488863331.1142858
l'écart type est : 22110.25398122522
le minimum de la série est : 524
le maximum de la série est : 99940
la somme des éléments de la série est : 268218
la médiane est : 5092.0

```

## Description De La Table By Région :

```
: data_grouped.describe()
```

	TotalPositiveCases	Deaths	Recovered	Active
count	21.000000	21.000000	21.000000	21.000000
mean	12772.285714	1689.380952	9930.285714	1152.619048
std	22110.253981	3692.874903	17038.680880	1574.390278
min	524.000000	23.000000	408.000000	23.000000
25%	2114.000000	146.000000	1442.000000	232.000000
50%	5092.000000	445.000000	4029.000000	509.000000
75%	11043.000000	1141.000000	8827.000000	1405.000000
max	99940.000000	16863.000000	76248.000000	6829.000000

- Dans cette description, nous allons faire une étude bivariée concernant les deux variables : "Recovered" et "Total Positive Cases"; Mettons la variable ' TotalPositiveCases ' variable dépendante et la variable 'Recovered' comme variable explicative.

## Calcul du matrice de covariance :

```

x = list(data["TotalPositiveCases"])
y = list(data["Recovered"])
np.cov(x,y)

array([[4.88863331e+08, 3.76359622e+08],
       [3.76359622e+08, 2.90316646e+08]])

```

### Pour le coefficient de corrélation :

```

np.corrcoef(x,y)

array([[1.          , 0.99901802],
       [0.99901802, 1.          ]])

```

- la donnée qui nous intéresse est donc en haut à droite et/ou en bas à gauche
- Plus le coefficient est proche de 1, la relation linéaire positive entre les variables est forte.
- Pour les coefficients de la droite des moindres carrés, on va utiliser la fonction linregress de scipy.stats.
- Elle prend en arguments les deux listes de données X et Y et renvoie 5 nombres réels. Seuls les 3 premiers nous intéressent : le premier est le coefficient directeur de la droite, le deuxième est l'ordonnée à l'origine et le troisième.

### Le coefficient de corrélation :

```

from scipy.stats import linregress
linregress(x,y)

LinregressResult(slope=0.7698667457966912, intercept=97.32767504300318, rvalue=0.9990180226756431, pvalue=3.467049988
4179204e-27, stderr=0.007832934981330633)

```

### Plot Nuage de points :

```

: plt.plot(x,y,'b*') #nuage de points
[<matplotlib.lines.Line2D at 0x7fd440c2fdd0>]



```

### Conversion de les données regroupées par RegionName (data) à un fichier (.csv) :

```

: data.to_csv('Data_Grouped.csv')

```

**Importation des données regroupées (Data\_Grouped) sur Logiciel R pour calculer l'équation de moindres carrées et affichées le plot de ligne de régression :**

**Importation des données regroupées (Data\_Grouped) :**

Untitled1\* Data\_Grouped Untitled2\*

Filter

X1	index	SNo	Date	Country	RegionCode	RegionName	Latitude	Longitude	Hos
1	0	3948	3948 2020-08-30 17:00:00	ITA	13	Abruzzo	42.35122	13.398438	
2	1	3949	3949 2020-08-30 17:00:00	ITA	17	Basilicata	40.63947	15.805148	
3	2	3950	3950 2020-08-30 17:00:00	ITA	18	Calabria	38.90598	16.594402	
4	3	3951	3951 2020-08-30 17:00:00	ITA	15	Campania	40.83957	14.250850	
5	4	3952	3952 2020-08-30 17:00:00	ITA	8	Emilia-Romagna	44.49437	11.341721	
6	5	3953	3953 2020-08-30 17:00:00	ITA	6	Friuli Venezia Giulia	45.64944	13.768136	
7	6	3954	3954 2020-08-30 17:00:00	ITA	12	Lazio	41.89277	12.483667	
8	7	3955	3955 2020-08-30 17:00:00	ITA	7	Liguria	44.41149	8.932699	
9	8	3956	3956 2020-08-30 17:00:00	ITA	3	Lombardia	45.46679	9.190347	
10	9	3957	3957 2020-08-30 17:00:00	ITA	11	Marche	43.61676	13.518875	
11	10	3958	3958 2020-08-30 17:00:00	ITA	14	Molise	41.55775	14.659161	
12	11	3959	3959 2020-08-30 17:00:00	ITA	21	P.A. Bolzano	46.49933	11.356624	
13	12	3960	3960 2020-08-30 17:00:00	ITA	22	P.A. Trento	46.06894	11.121231	
14	13	3961	3961 2020-08-30 17:00:00	ITA	1	Piemonte	45.07327	7.680687	

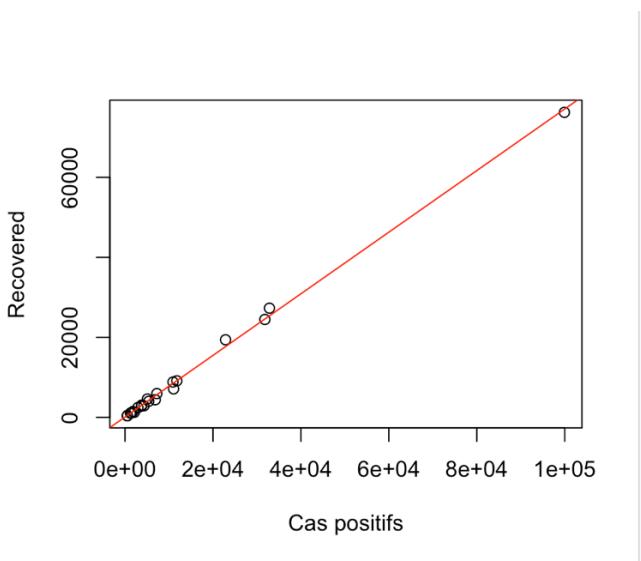
Showing 1 to 14 of 21 entries

### Plot nuage des points avec linregress :

```

1 reg <- lm(Recovered~ TotalPositiveCases, data = Data_Grouped)
2 summary(reg)
3 plot(Data_Grouped$TotalPositiveCases,Data_Grouped$Recovered , ylab = "Recovered", xlab = "Cas positifs")
4 abline(reg, col = "red")
5

```



### Calcul des deux variables a et b de l'équation de moindres carrés :

```

1 COV = cov(Data_Grouped$Recovered,Data_Grouped$TotalPositiveCases)
2 VAR = var(Data_Grouped$TotalPositiveCases)
3 a = COV/VAR
4 a
5 b = mean(Data_Grouped$Recovered)-a*mean(Data_Grouped$TotalPositiveCases)
6 b

```

```

> COV = cov(Data_Grouped$Recovered,Data_Grouped$TotalPositiveCases)
> VAR = var(Data_Grouped$TotalPositiveCases)
> a = COV/VAR
> a
[1] 0.7698667
> b = mean(Data_Grouped$Recovered)-a*mean(Data_Grouped$TotalPositiveCases)
> b
[1] 97.32768

```

on peut déduire que parmi les x cas positifs il y'en a y guérisons selon la méthode des moindres carrées.

## Partie 4 : Conclusion

- Les premiers cas de coronavirus à propagation terrestre en Italie sont apparus dans les régions du nord de la Lombardie, de la Vénétie et de l'Émilie-Romagne le 20 février
- La collecte des données a commencé le 24 février
- Le 8 mars 2020, le Premier ministre Giuseppe Conte a étendu la quarantaine à toute la Lombardie et à 14 autres provinces du nord, et le lendemain à toute l'Italie, plaçant plus de 60 millions de personnes en quarantaine.
- Le 11 mars 2020, PM Conte a interdit la quasi-totalité des activités commerciales à l'exception des supermarchés et des pharmacies.
- Le 16 mars 2020, l'Italie est devenue le centre mondial des cas actifs de coronavirus avec deux fois plus de cas actifs de tout autre pays, y compris la Chine et l'Iran, combinés à 20603 cas actifs. Les USA ont pris le relais quelques semaines plus tard, le 11 avril.
- Au 8 mai 2020, l'Italie comptait 87 961 cas actifs, l'un des nombres les plus élevés au monde. Dans l'ensemble, il y a eu 217 185 cas confirmés et 30 201 décès (un taux de mortalité d'environ 500 par million d'habitants), tandis qu'il y a eu 99 023 récupérations ou licenciements.
- Le 8 mai, l'Italie avait testé environ 1 610 000 personnes.
- Après 30 août et d'après analyse des données de ce projet on conclue que l'Italie dans une situation un peu stable car on observe que les personnes récupérés sont plus nombreux que les personnes mourir et tout cela grâce à les décisions de l'état et le gouvernement (par exemple : Confinement, Les règles sanitaires, Les efforts des hôpitaux )

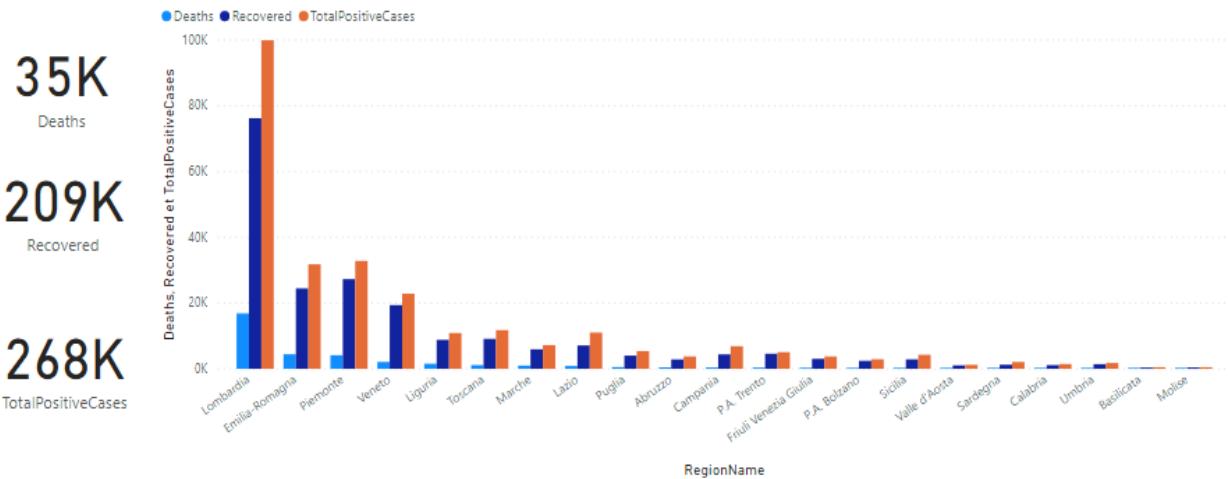
## Partie 5 : Dashboard



# Coronavirus in Italy

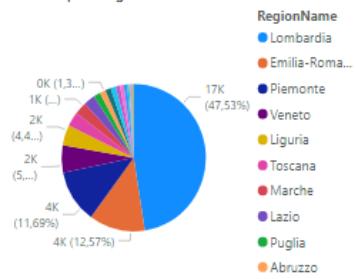


Deaths, Recovered et TotalPositiveCases par RegionName

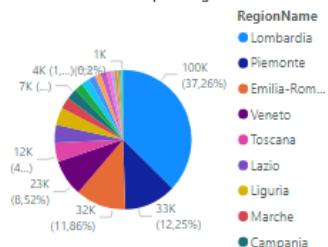


Réalisé par :  
Yassin Latif  
Mourad Ezzebdi

Deaths par RegionName

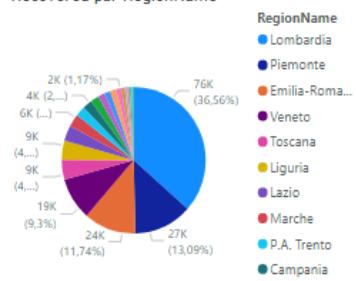


TotalPositiveCases par RegionName

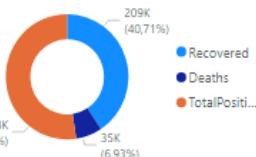


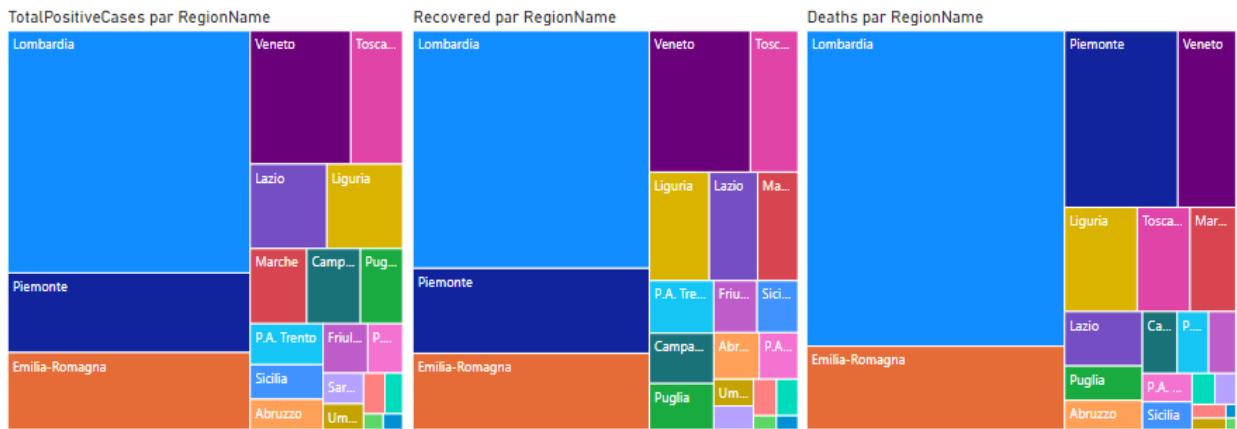
RegionName	TotalPositiveCases	Recovered	Deaths
Abruzzo	3773	2872	472
Basilicata	524	408	28
Calabria	1477	1148	97
Campania	6882	4412	445
Emilia-Romagna	31805	24478	4459
Friuli Venezia Giulia	3764	3056	348
Lazio	11043	7130	878
Liguria	10907	8827	1571
Lombardia	99940	76248	16863
Marche	7238	5949	987
Molise	525	432	23
P.A. Bolzano	2932	2450	292
P.A. Trento	5092	4600	405
Piemonte	32844	27293	4146
Puglia	5402	4029	556
Sardegna	2114	1268	134
Sicilia	4291	2891	286
Toscana	11785	9141	1141
Umbria	1784	1442	80
Valle d'Aosta	1232	1063	146
Veneto	22864	19399	2120
<b>Total</b>	<b>268218</b>	<b>208536</b>	<b>35477</b>

Recovered par RegionName



Recovered, Deaths et TotalPositiveCases





Deaths, Recovered et TotalPositiveCases par RegionName

