How is our cities' climate going to change in the future ?

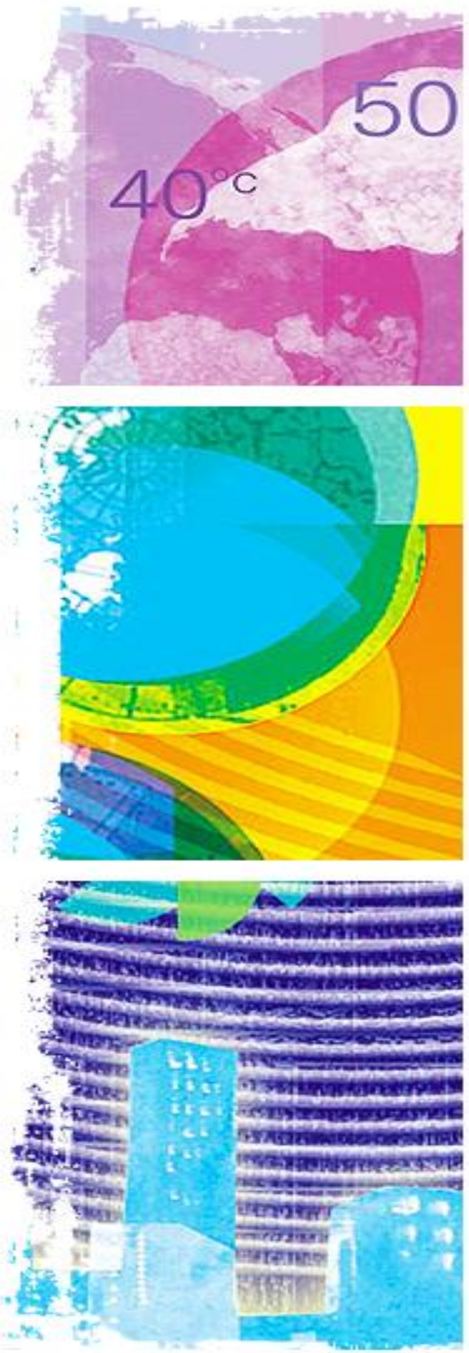
Kaci BOURGUA

Data Analytics, March 2023

Content of this presentation

1. Introduction
2. Data Sources, Extraction & Cleaning
3. Exploratory Data Analysis & Visualisation
4. Database & SQL Queries
5. Machine Learning Models
6. Demo & Insights

1. Introduction



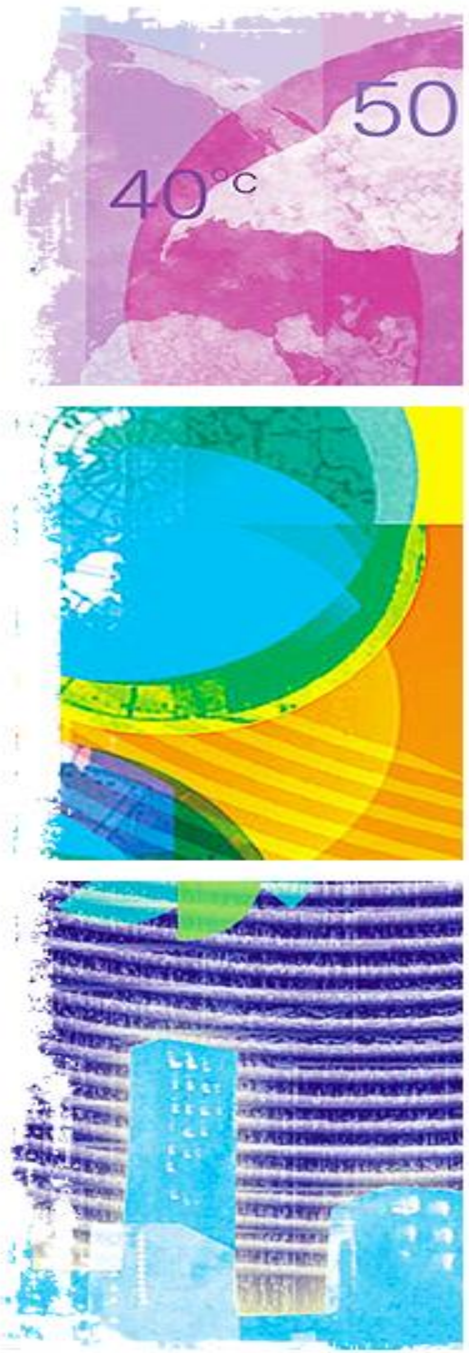
Context & Business Case

Personal interest in environmental topics and sustainability-oriented innovations

Private companies

Public policy makers and city administrations

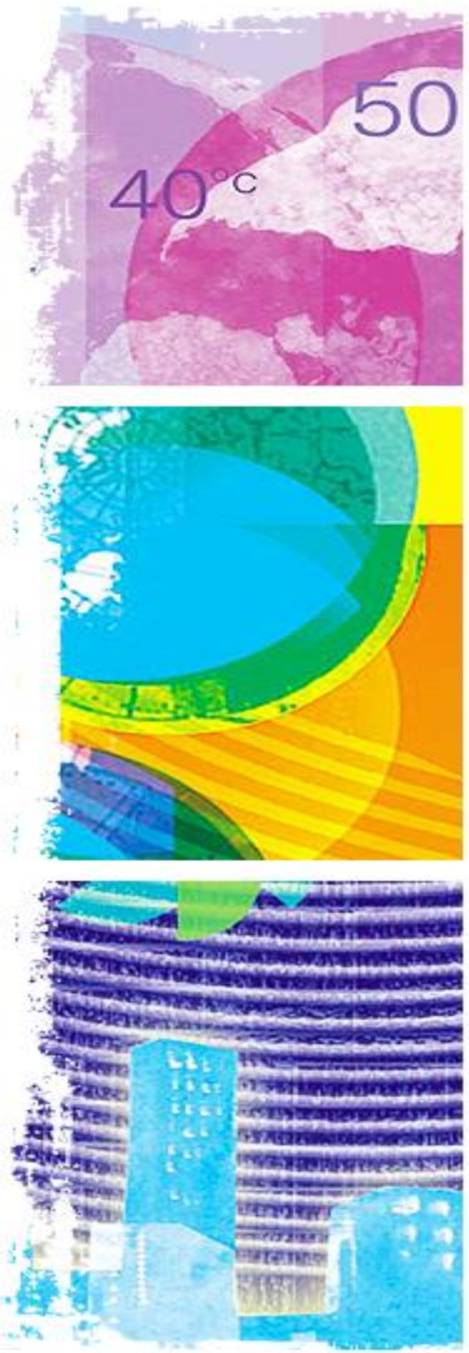
How is our cities' climate going to change in the future ?



Planning

1. Get the right data from Copernicus & extract it
2. Data cleaning, preparation & exploratory data analysis
3. Database creation, ERD & queries on MySQL
4. Testing supervised ML models with time series analysis
5. Demo with the best model and comparison with forecast data

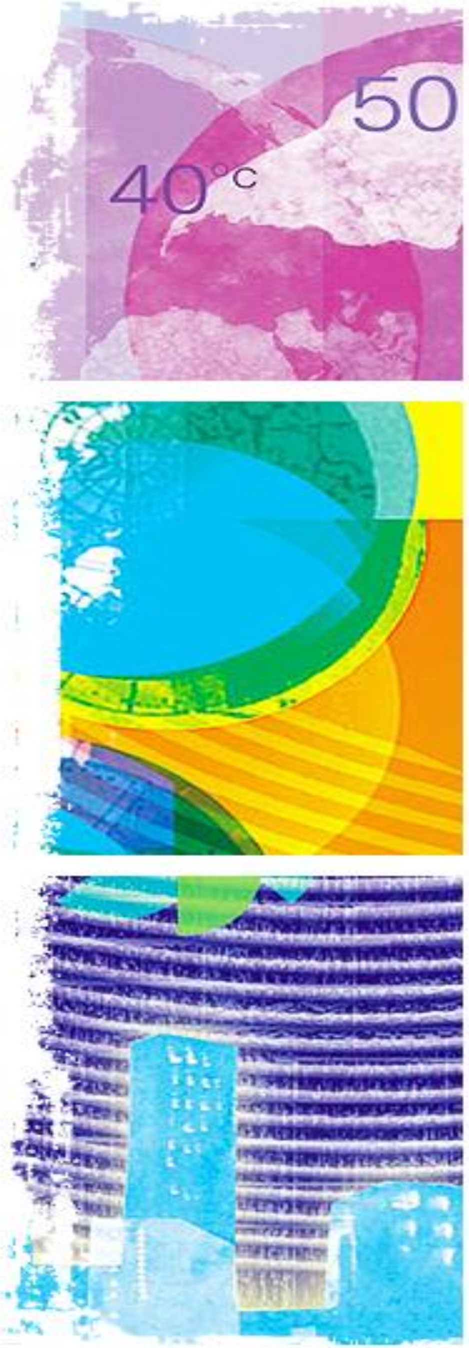
2. Data Sources, Extraction & Cleaning



Data Sources

- European Union's Earth Observation Program
- Satellite missions & ground-based monitoring systems for climate data
- Global bioclimatic indicators from 1950 to 2100 derived from climate projection
- RCP8.5 scenario

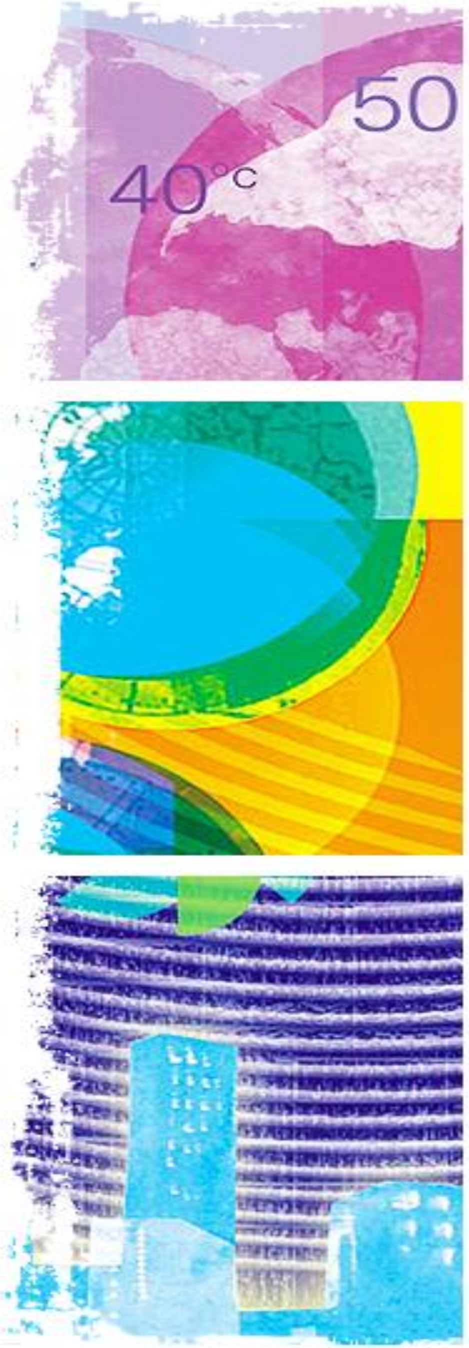
- Geographical Database
- Dataset of all the cities around the world with more than 1000 inhabitants
- Features : city name, country name, population and latitude/longitude
- Used to extract the data for my analysis



Features

Variable Name	Unit	Description
Annual mean temperature (BIO01)	K (Converted to °C)	Annual mean of the daily mean temperature at 2 m above the surface. This indicator corresponds to the official BIOCLIM variable BIO01 that is used in ecological niche modelling.
Mean temperature of warmest quarter (BIO10)	K (Converted to °C)	The mean of monthly mean temperature during the warmest quarter, defined as the quarter with the highest monthly mean (of the daily mean) temperature using a moving average of 3 consecutive months. This indicator corresponds to the official BIOCLIM variable BIO10.
Mean temperature of coldest quarter (BIO11)	K (Converted to °C)	The mean of monthly mean temperature during the coldest quarter, defined as the quarter with the lowest monthly mean (of the daily mean) temperature using a moving average of 3 consecutive months. This indicator corresponds to the official BIOCLIM variable BIO11.
Annual precipitation (BIO12)	m s-1	Annual mean of the daily mean precipitation rate (both liquid and solid phases). This indicator corresponds to the official BIOCLIM variable BIO12. To compute the total precipitation sum over the year, a conversion factor should be applied of 3600x24x365x1000 (mm year-1).
Precipitation of wettest month (BIO13)	m s-1	Maximum of the monthly precipitation rate. To compute the total precipitation sum over the month, a conversion factor should be applied of 3600x24x30.4 (average number of days per month)x1000. This indicator corresponds to the official BIOCLIM variable BIO13.
Precipitation of driest month (BIO14)	m s-1	Minimum of the monthly precipitation rate. To compute the total precipitation sum over the month, a conversion factor should be applied of 3600x24x30.4 (average number of days per month)x1000. This indicator corresponds to the official BIOCLIM variable BIO14.

Data Cleaning (cities dataset)




```python
df_city.columns = df_city.columns.str.lower()
df_city.columns = df_city.columns.str.replace(' ','_')
```

Python

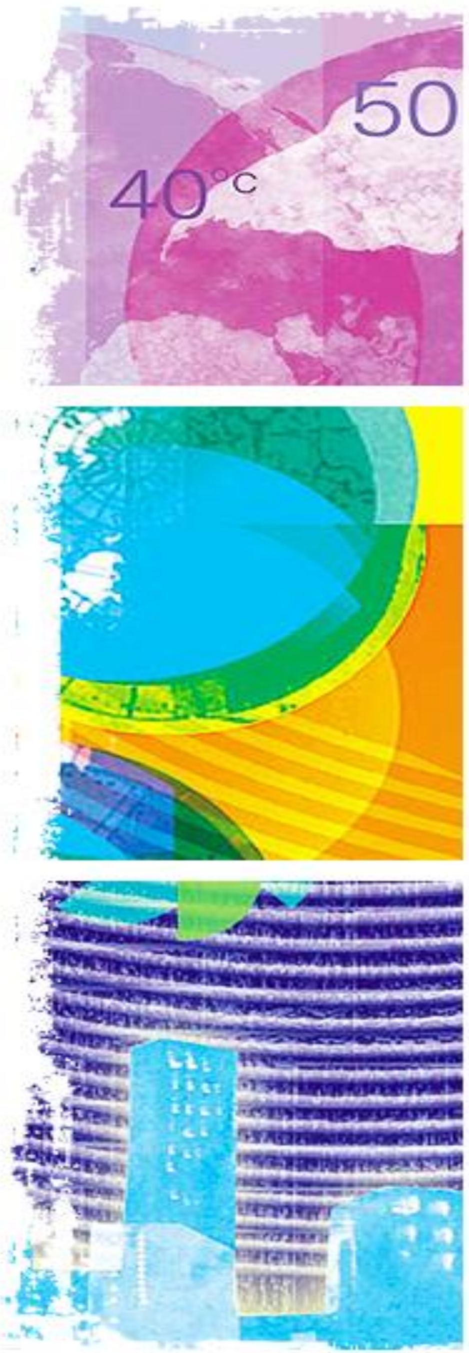
```python
df_city[['latitude', 'longitude']] = df_city['Coordinates'].str.split(',', 1, expand=True)
```

```python
df_city.drop(['alternate_names', 'country_code_2', 'admin2_code', 'admin3_code', 'admin4_code'
```

Python

```python
df.drop(columns=['Unnamed: 0', 'ascii_name', 'feature_class', 'feature_code', 'admin1_code', '
```

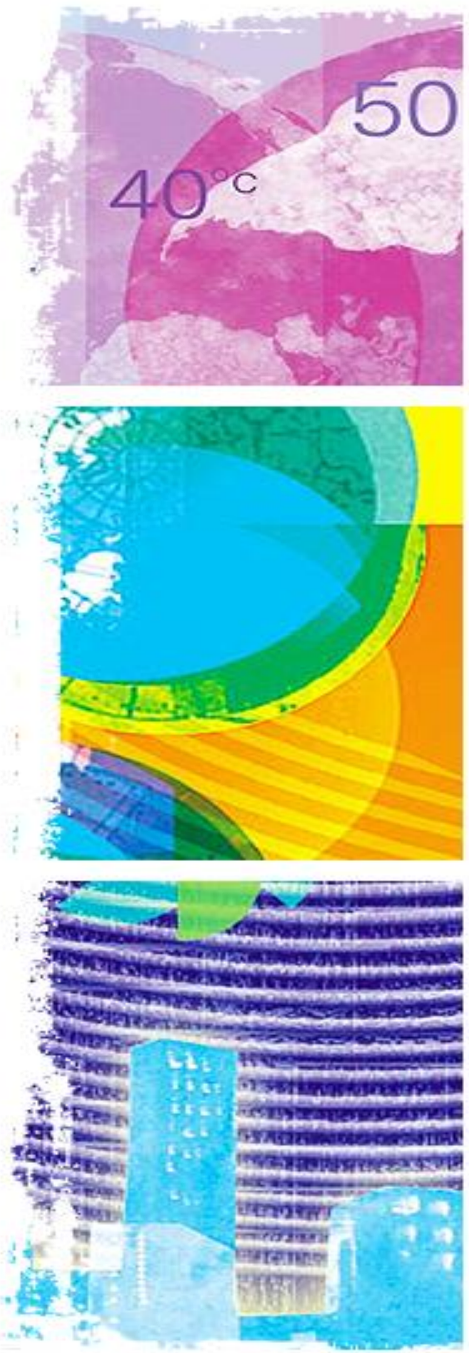
Python



Data Extraction

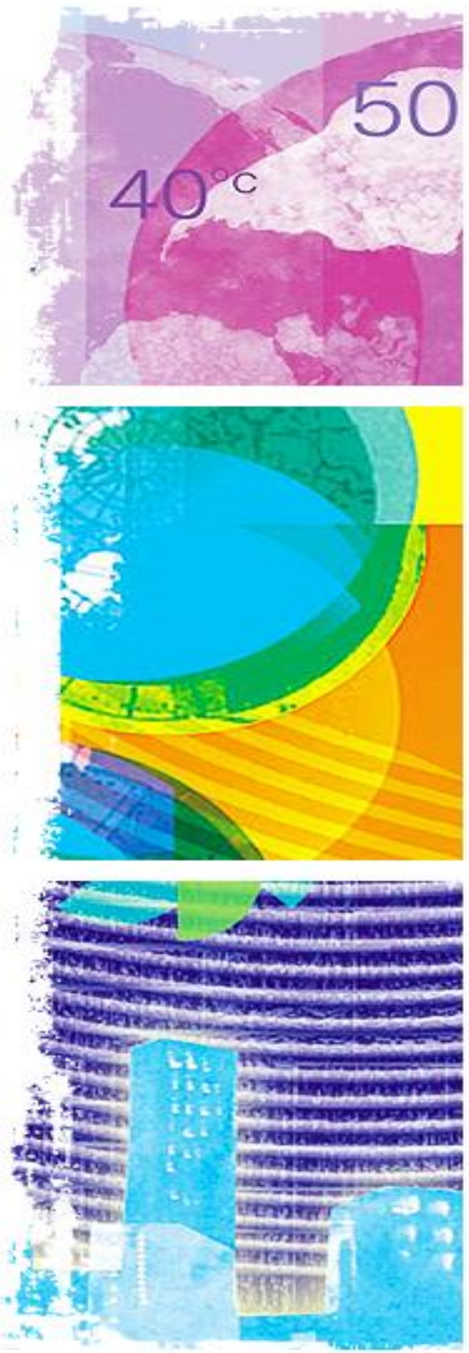
Extraction of the values of one indicator for 100 cities, from NetCDF files

```
import netCDF4
import pandas as pd
import numpy as np
import glob


# Record all the years of the netCDF files into a Python list

data = netCDF4.Dataset('dataset-sis-biodiversity/BIO14_ipsl-cm5a-lr_rcp85_r1i1p1_1950-2100_v1.
TIME = data.variables["time"]
LAT=data.variables["latitude"][:]
LON=data.variables["longitude"][:]
INDICATOR=data.variables["BIO14"]

# Creating an empty Pandas DataFrame covering the whole range of data

starting_date = '1950'
ending_date = '2100'
date_range = pd.date_range(start=starting_date, end=ending_date, freq='AS')
dt = np.arange(0, data.variables["time"].size)

df = pd.DataFrame(columns=['year','geoname_id','precipitation_driest_month'],index=date_range)
dt = np.arange(0, data.variables["time"].size)
```



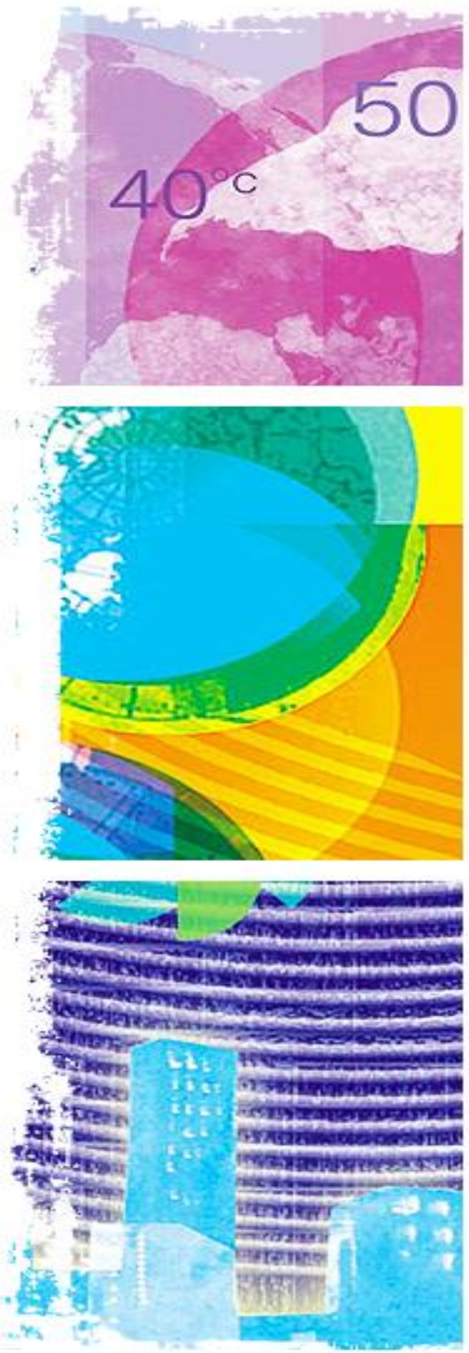
Data Extraction

```python
# Defining the location, lat, lon based on the csv data
cities = pd.read_csv('top_100_cities.csv')

for index, row in cities.iterrows():
    location = row['name']
    location_latitude = row['latitude']
    location_longitude = row['longitude']
    city_id=row['geoname_id']

    # Squared difference between the specified lat,lon and the lat,lon of the netCDF
    sq_diff_lat = (LAT - location_latitude)**2
    sq_diff_lon = (LON - location_longitude)**2

    # Identify the index of the min value for lat and lon
    min_index_lat = sq_diff_lat.argmin()
    min_index_lon = sq_diff_lon.argmin()


    for time_index in dt:
        df.iloc[time_index] = [df.index[time_index], city_id,INDICATOR[time_index, min_index_l

    print('Recording the value for '+ location)
    df.to_csv(fr"6.precipitation_driest_month/{location}.csv")
```



Data Extraction

```
final_df6 = pd.concat(map(pd.read_csv, glob.glob(fr'6.precipitation_driest_month/*.csv')))

final_df6.drop('Unnamed: 0', axis=1, inplace=True)

final_df6['year']=final_df6['year'].str.split('-').str[0]

final_df6.to_csv('6.precipitation_driest_month.csv', index=None)
```

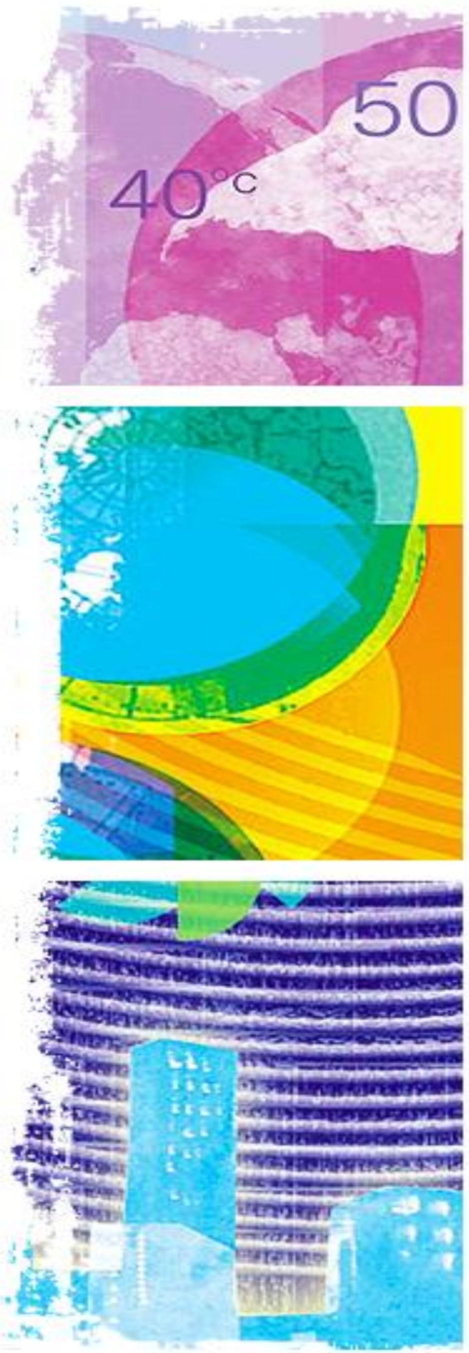


Data Cleaning (bioclimatic indicators)

```python
#Converting kelvin to celcius
df['annual_mean_temperature']=df['annual_mean_temperature'].apply(lambda x:x-273,15)
df['mean_temperature_coldest_quarter']=df['mean_temperature_coldest_quarter'].apply(lambda x:x
df['mean_temperature_warmest_quarter']=df['mean_temperature_warmest_quarter'].apply(lambda x:x
```

Python

Liveability Index



Annual Mean Temperature — 20%

Mean Temperature Coldest Quarter — 30%

Mean Temperature Warmest Quarter — 30%

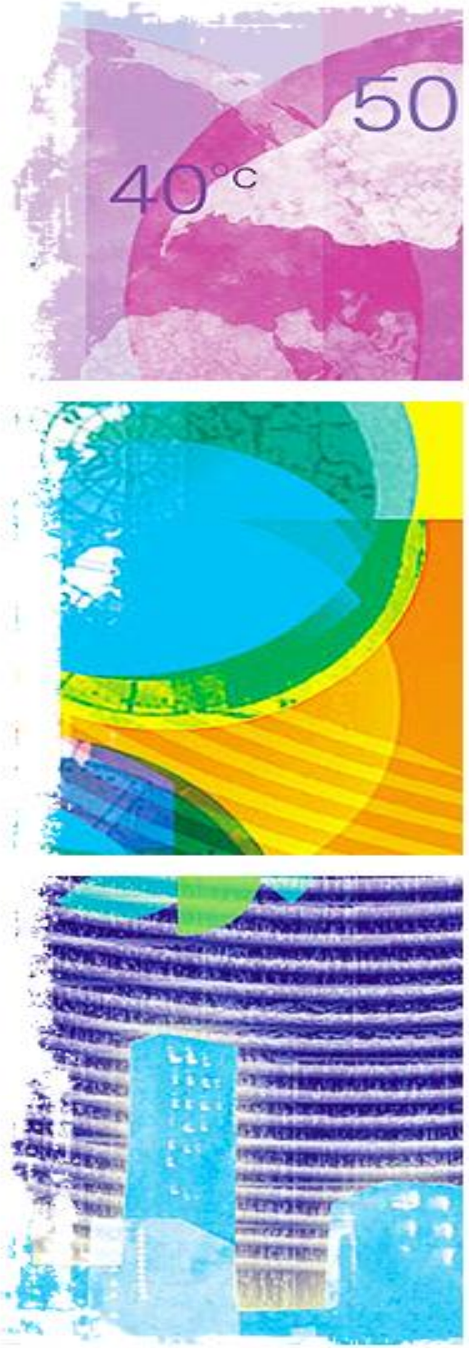
Annual Mean Precipitation — 20%

Mean Precipitation Wettest Month — 5%

Mean Precipitation Driest Month — 5%

- Index between 0 & 1
- The lower, the better

Index creation

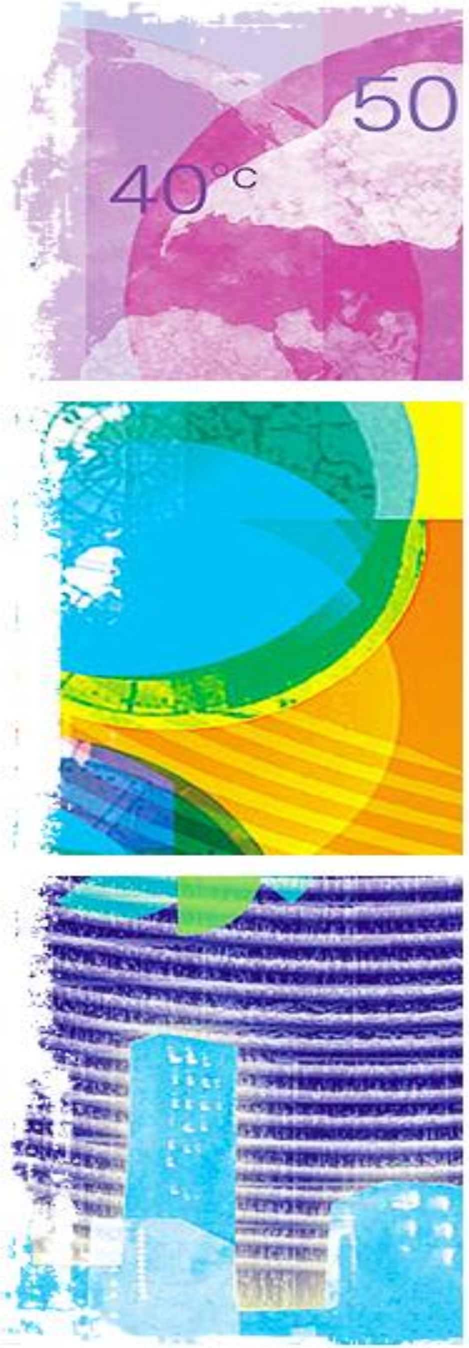



```
from sklearn.preprocessing import MinMaxScaler

# Define the weights for each climate score variable
climate_weights = {
    'annual_mean_temperature': 0.2,
    'mean_precipitation': 0.1,
    'mean_temperature_coldest_quarter': 0.3,
    'mean_temperature_warmest_quarter': 0.3,
    'precipitation_driest_month': 0.05,
    'precipitation_wettest_month': 0.05
}

# Load the data into a pandas dataframe
dfx = pd.read_csv('full_data_top_100_cities_celcius_rcp8.5.csv')

# Normalize the climate score variables using MinMaxScaler
scaler = MinMaxScaler()
dfx[list(climate_weights.keys())] = scaler.fit_transform(dfx[list(climate_weights.keys())])

# Multiply each climate score variable by its corresponding weight
weighted_scores = dfx[list(climate_weights.keys())].multiply(list(climate_weights.values()))

# Sum up the weighted scores to get the weighted index for the climate score
dfx['climate_score_weighted'] = weighted_scores.sum(axis=1)
dfx
```

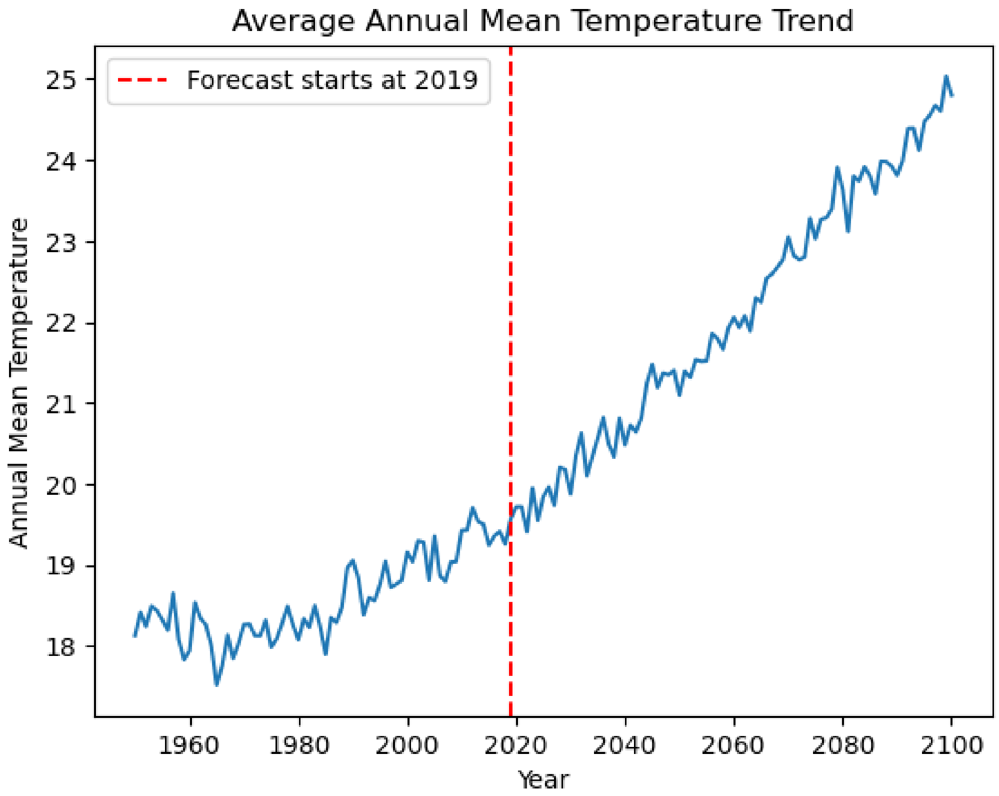
Scale the data

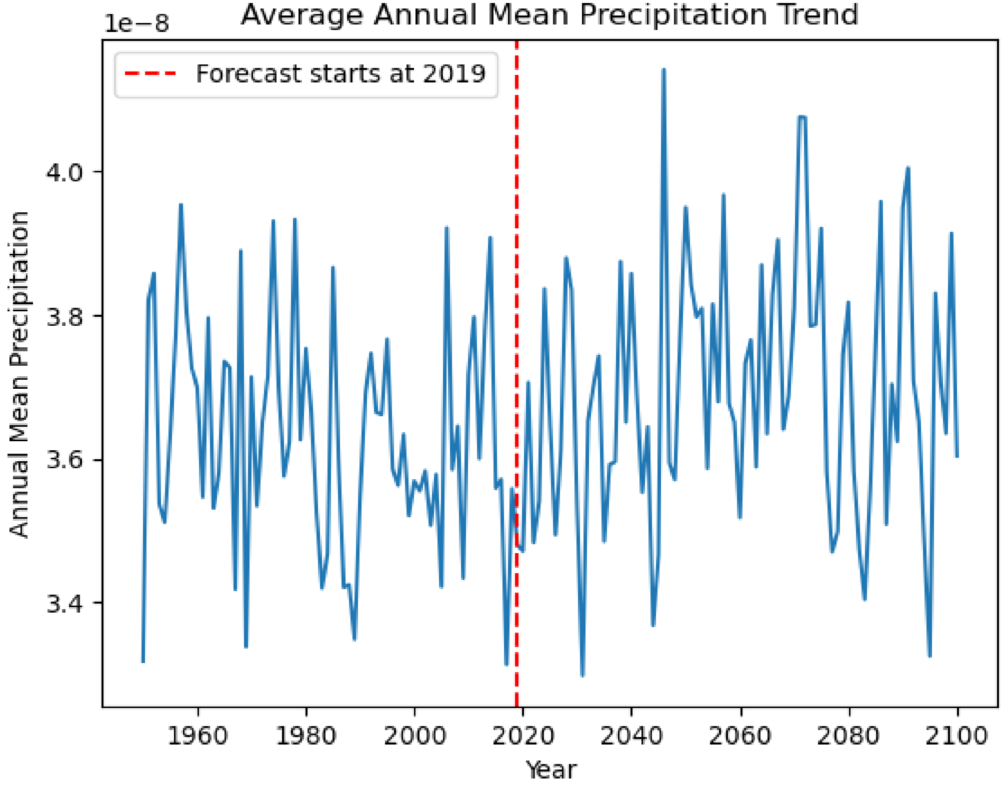
Assign a weight

3. Exploratory Data Analysis & Visualisation



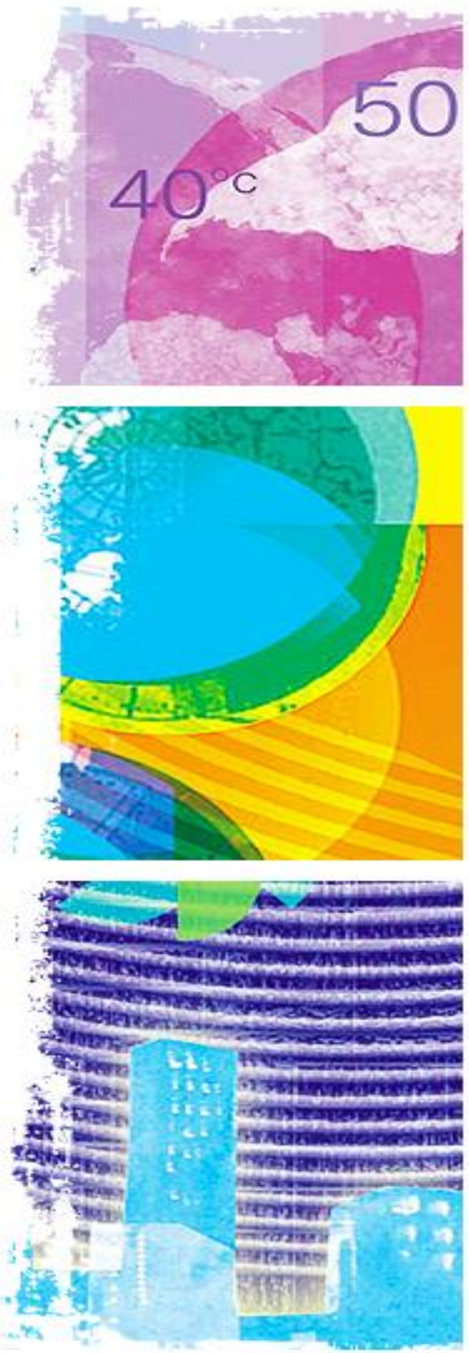
Exploratory Data Analysis





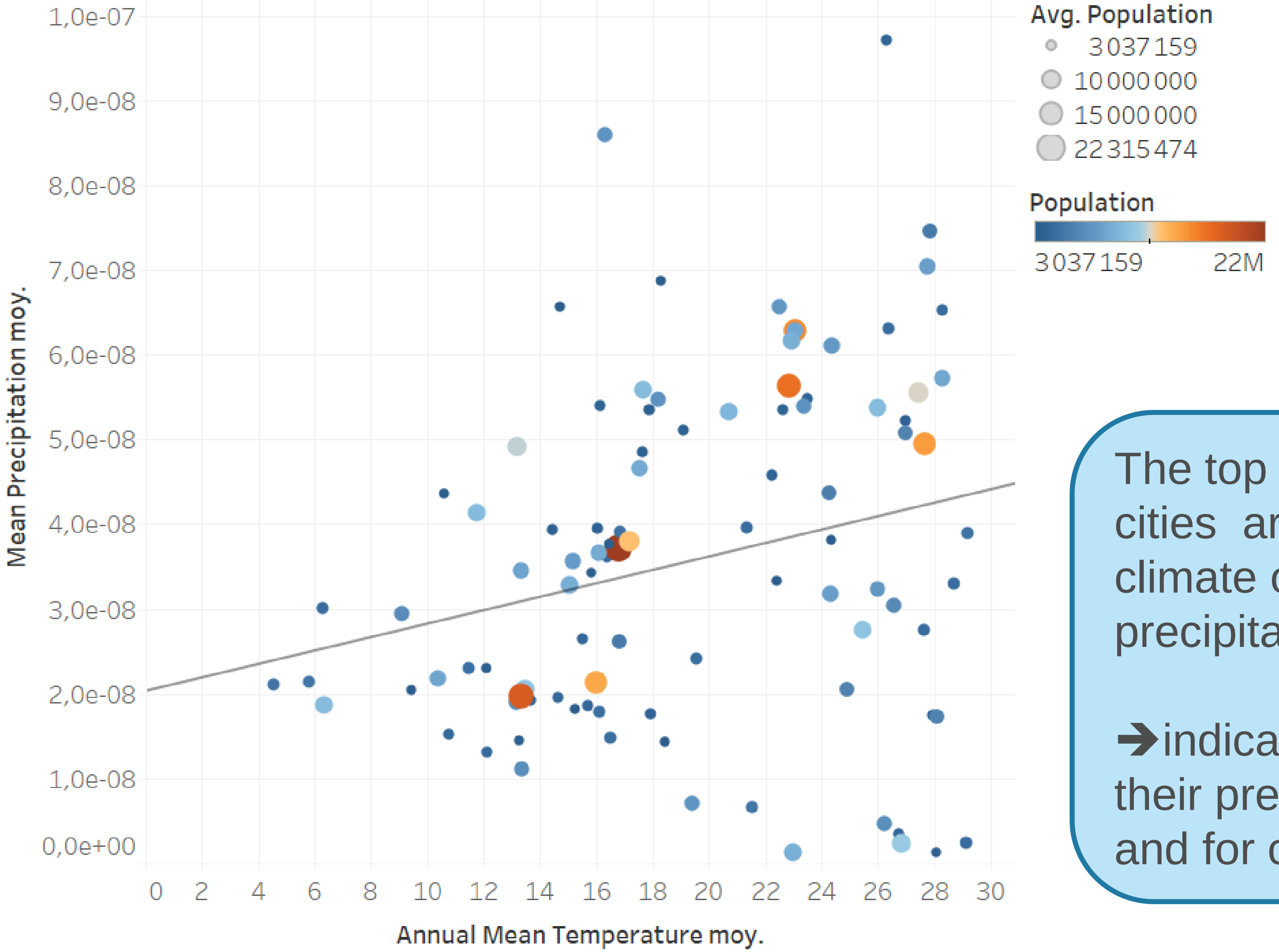
- The RCP 8.5 forecast shows a steep increase in average annual temperature for the 100 cities
- For precipitation, the trend is less clear, and seems to follow a cyclic pattern. However, we see more extreme values

Exploratory Data Analysis



Temperature & Precipitation Scatter Plot

* average values for 100 cities, for 2018



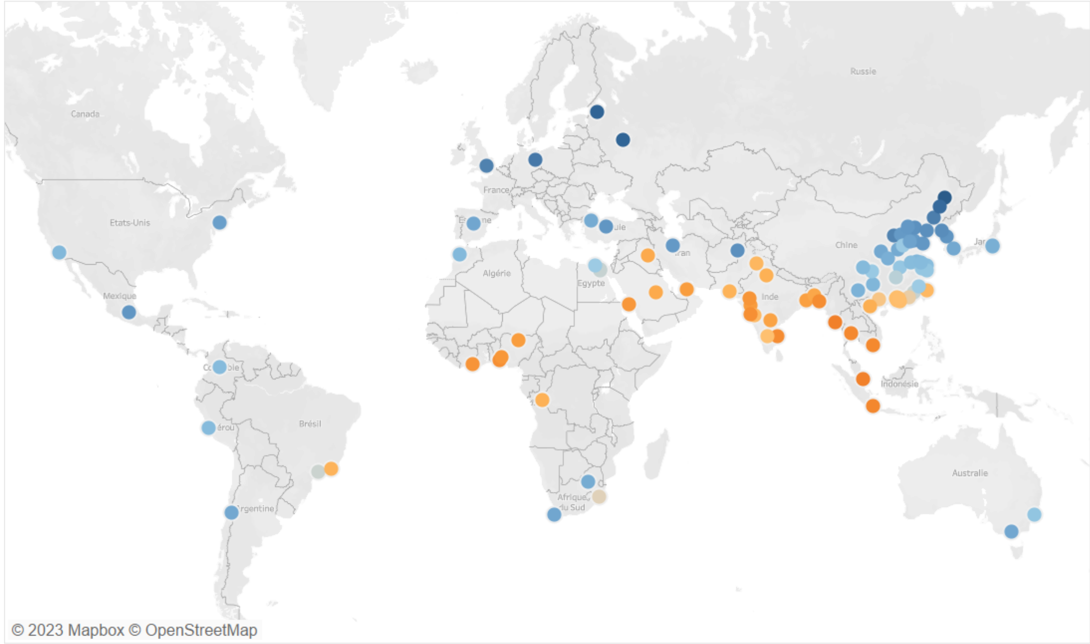
Average of Annual Mean Temperature vs. average of Mean Precipitation. Color shows average of Population. Size shows average of Population. Details are shown for Name. The data is filtered on Year, which ranges from 2018 to 2018.

The top 100 most populated cities are cities with warm climate on average, that receive precipitation

➔indicates that humans have their preferences for climate, and for cities to thrive

Exploratory Data Analysis

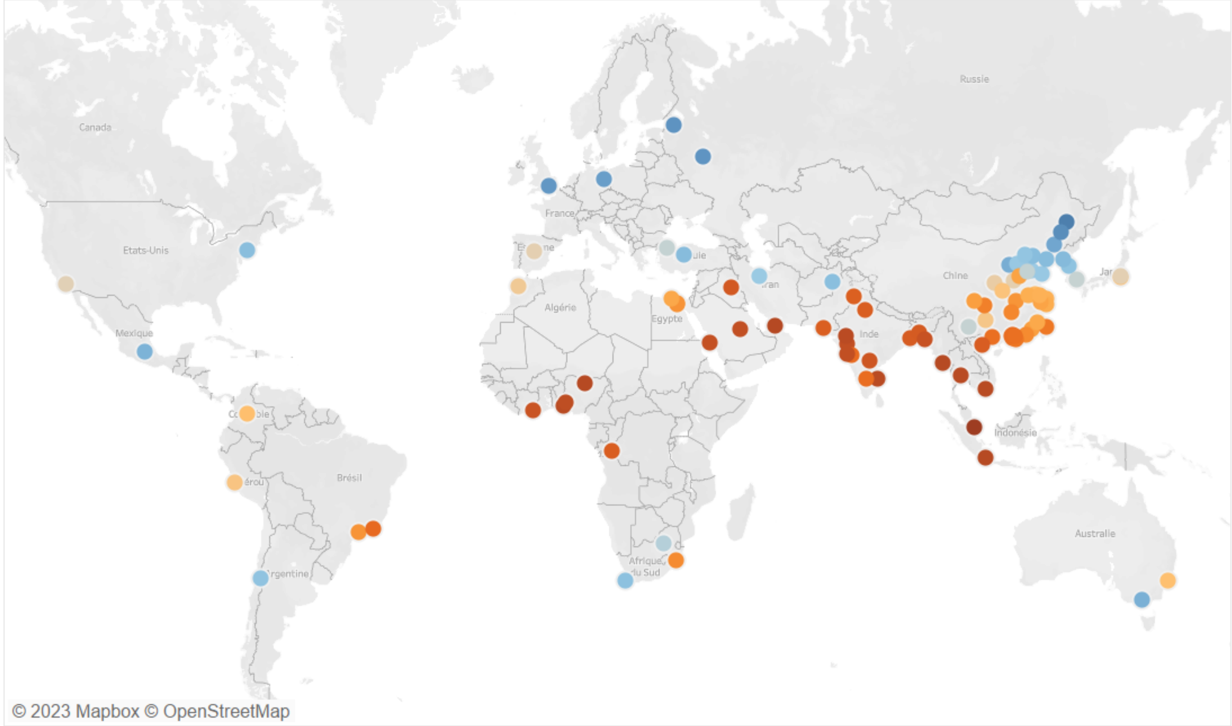
Liveability Index for 2018



Avg. Climate Li..
0,1282 0,6517

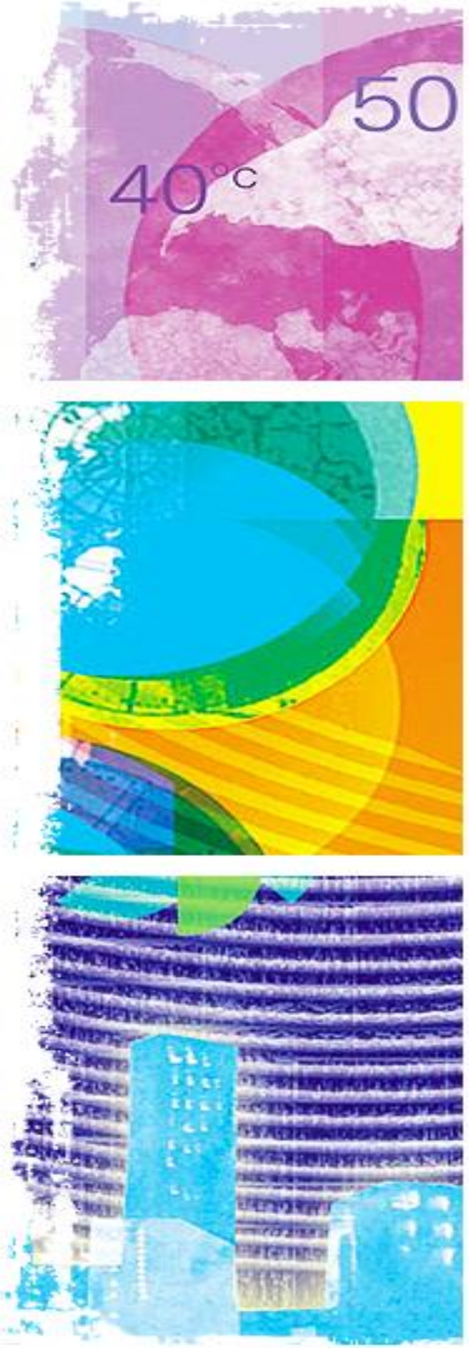
Map based on Longitude (generated) and Latitude (generated). Color shows average of Climate Liveability Index. Details are shown for Name and Country Name En. The data is filtered on Year, which ranges from 2018 to 2018.

Liveability Index for 2100

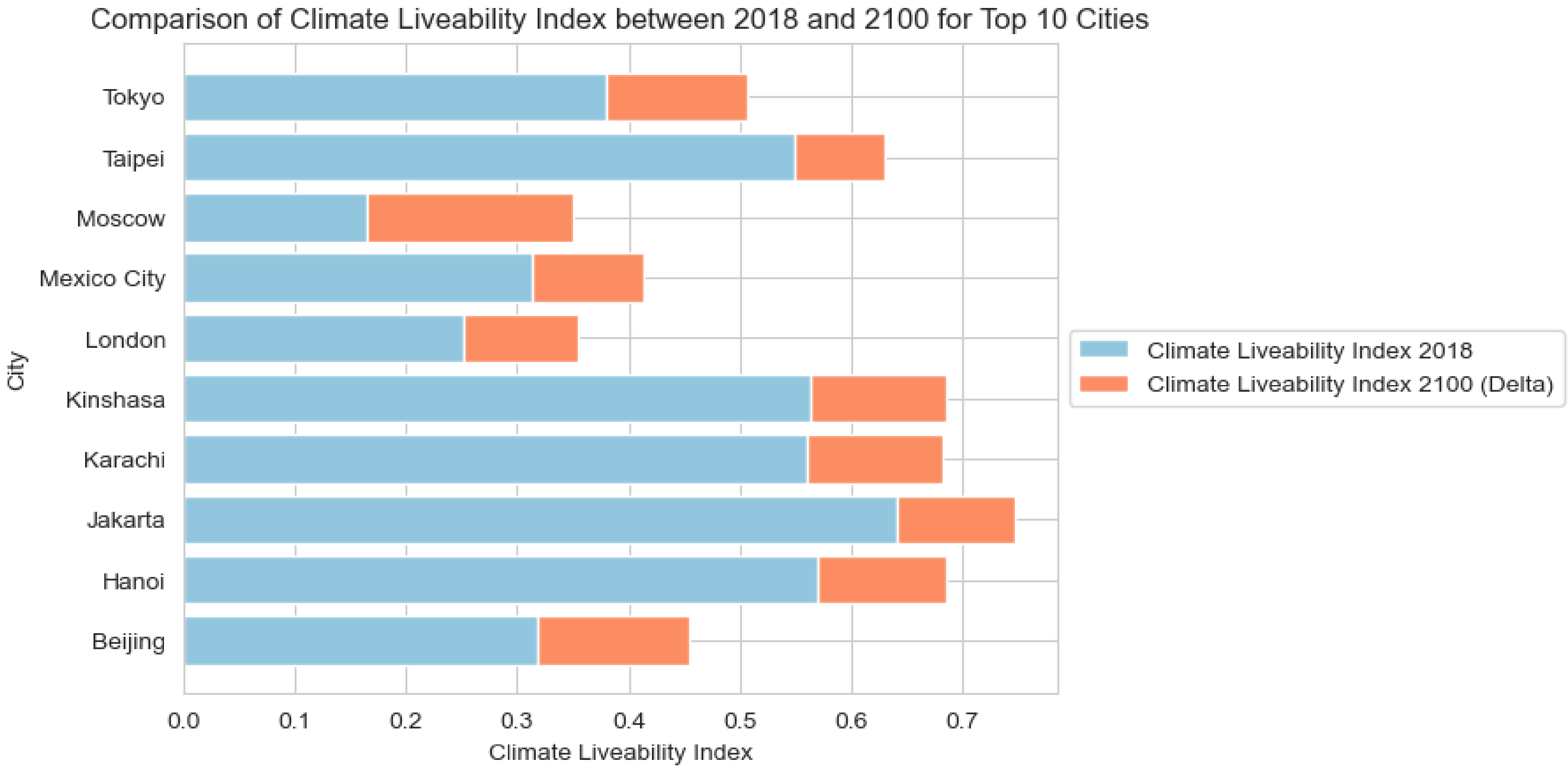


Avg. Climate Li..
0,2987 0,7905

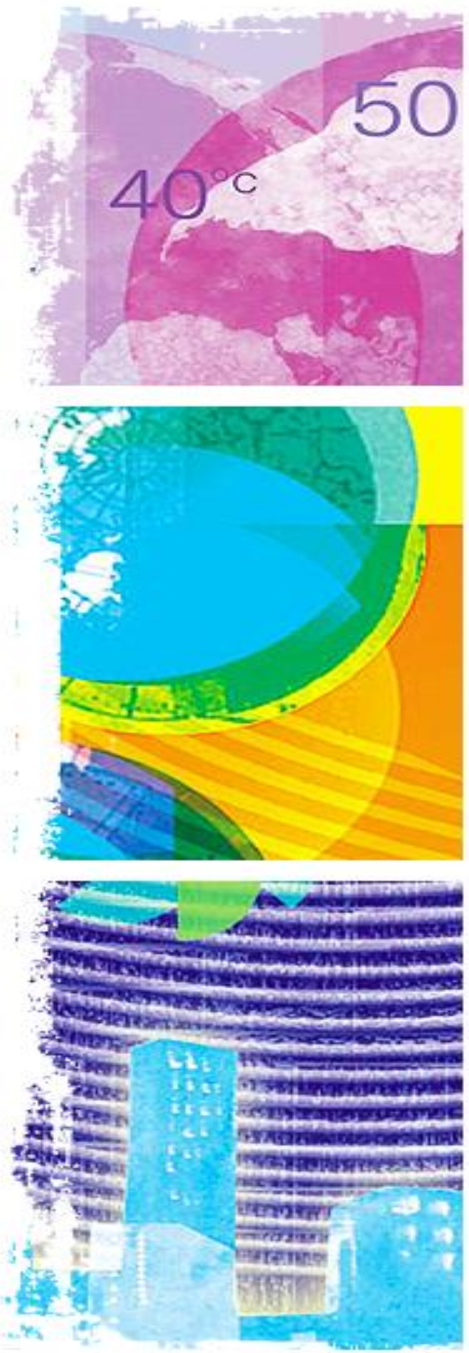
Map based on Longitude (generated) and Latitude (generated). Color shows average of Climate Liveability Index. Details are shown for Name and Country Name En. The data is filtered on Year, which ranges from 2100 to 2100.



Exploratory Data Analysis



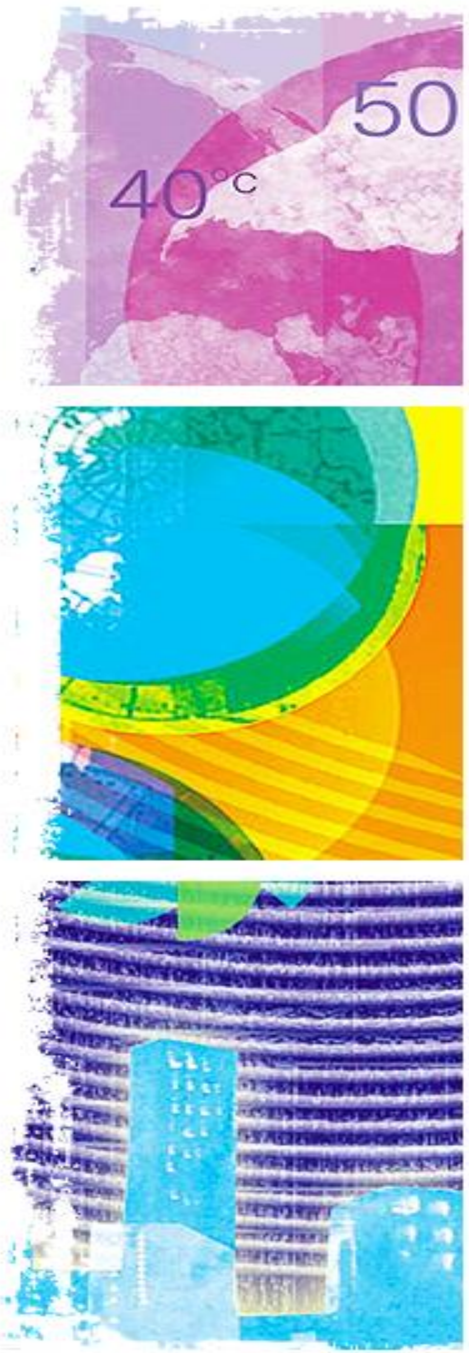
4. Database & SQL Queries



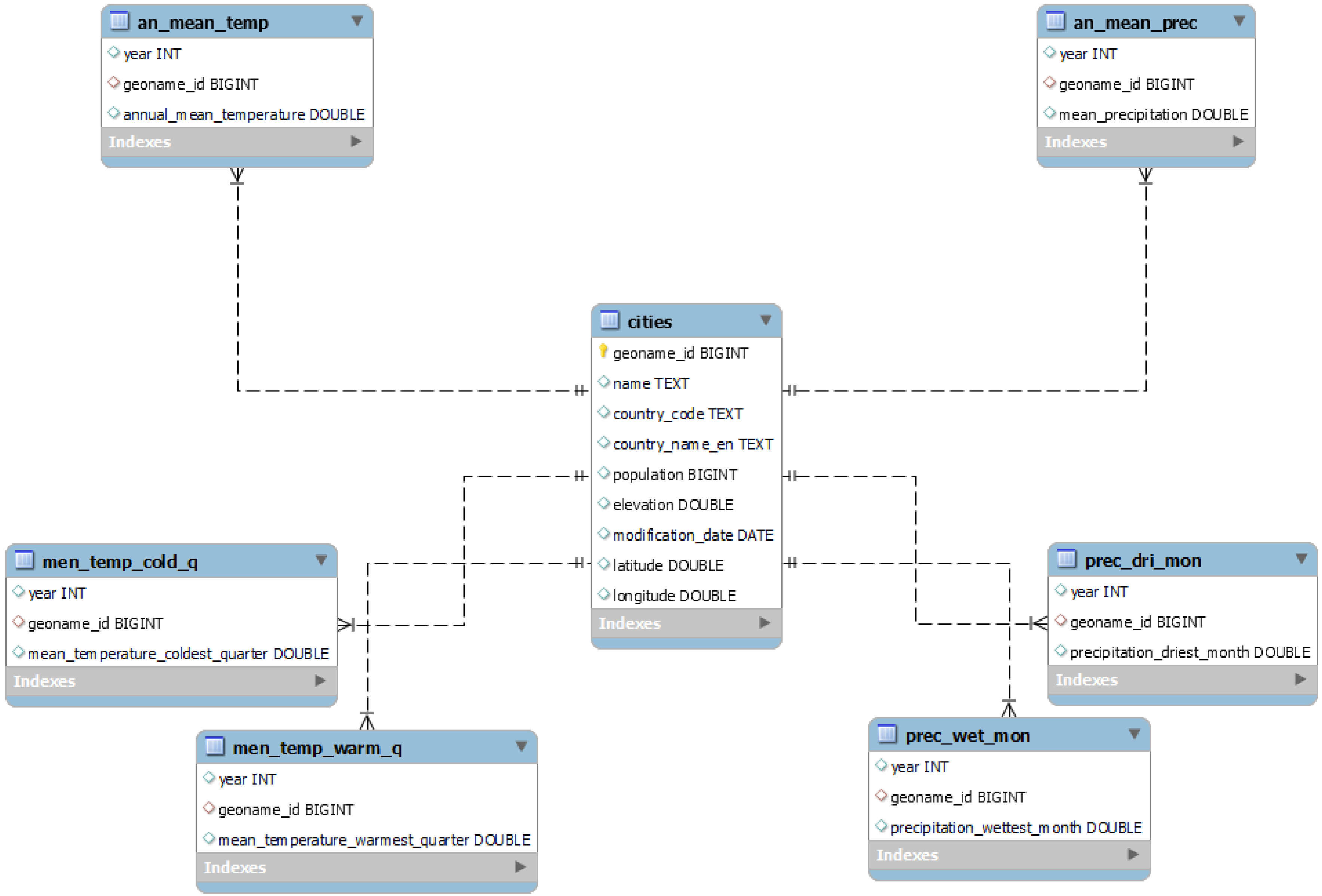
Database creation

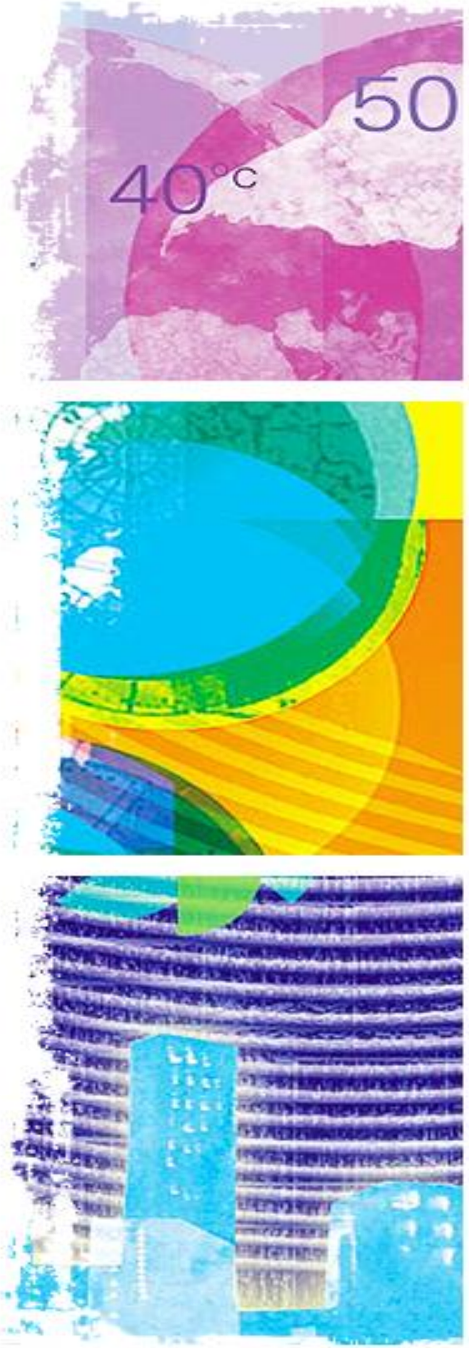
I have created my database using the Python library sqlAlchemy with the code bellow

```
from sqlalchemy import create_engine
import pymysql.cursors
import os
import getpass
pw = getpass.getpass()
connection_string = 'mysql+pymysql://root:' + pw + '@localhost:3306/'
engine = create_engine(connection_string)

#Create each table from my pandas dataframes
df_cities.to_sql('cities', engine, 'climate_fproject', if_exists='replace', index = False)
annual_mean_temp_df.to_sql('an_mean_temp', engine, 'climate_fproject', if_exists='replace', index = False)
mean_temp_warmest_quarter_df.to_sql('men_temp_warm_q', engine, 'climate_fproject', if_exists='replace', index = False)
mean_temp_coldest_quarter_df.to_sql('men_temp_cold_q', engine, 'climate_fproject', if_exists='replace', index = False)
annual_mean_precipitation_df.to_sql('an_mean_prec', engine, 'climate_fproject', if_exists='replace', index = False)
precipitation_wettest_month_df.to_sql('prec_wet_mon', engine, 'climate_fproject', if_exists='replace', index = False)
precipitation_driest_month_df.to_sql('prec_dri_mon', engine, 'climate_fproject', if_exists='replace', index = False)
```



Entity Relationship Diagram





SQL Queries

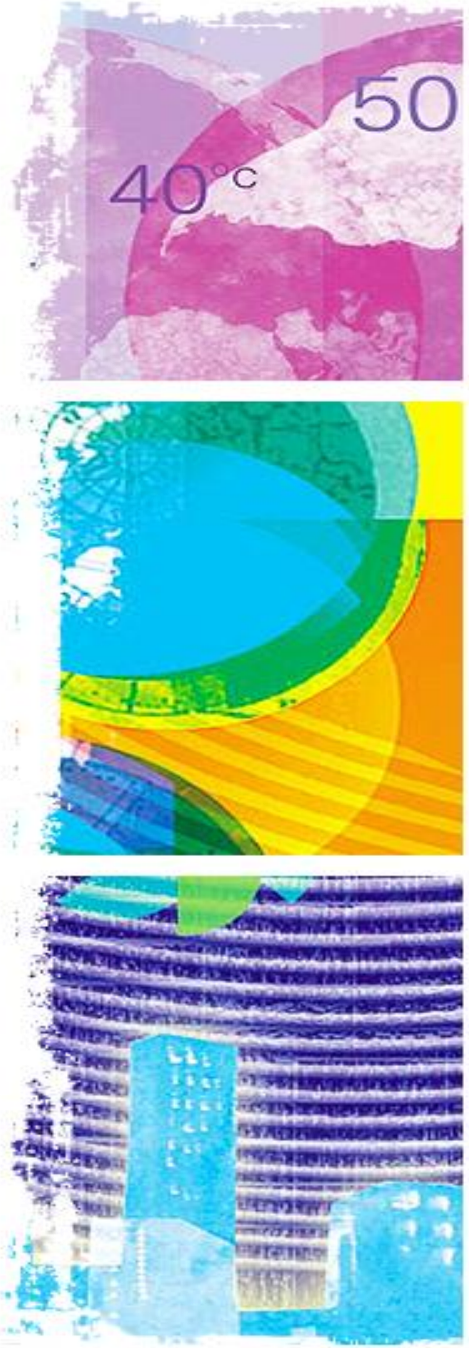
```sql
-- Query 2 : Top 50 cities in the World, by population
select name, population, country_name_en
from cities
order by population desc
limit 50;
```

name	population	country_name_en
Shanghai	22315474	China
Beijing	18960744	China
Shenzhen	17494398	China
Guangzhou	16096724	China
Lagos	15388000	Nigeria
Istanbul	14804116	Turkey
Chengdu	13568357	China
Mumbai	12691836	India
Mexico City	12294193	Mexico

```sql
-- Query 4 : Average mean temperature (with convertion in Celcius) for the top 30 cities in the World from 1950 to 2100
select amt.geoname_id, c.name, amt.annual_mean_temperature,
round((amt.annual_mean_temperature - 273.15), 2) as temperature_in_celcius,amt.year
from cities c
right join an_mean_temp amt
on c.geoname_id=amt.geoname_id
order by name asc, year asc;
```

geoname_id	name	annual_mean_temperature	temperature_in_celcius	year
2293538	Abidjan	298.82852	25.68	1950
2293538	Abidjan	298.45282	25.3	1951
2293538	Abidjan	298.19434	25.04	1952
2293538	Abidjan	298.4637	25.31	1953
2293538	Abidjan	298.43268	25.28	1954
2293538	Abidjan	298.13373	24.98	1955
2293538	Abidjan	298.47598	25.33	1956
2293538	Abidjan	298.37076	25.22	1957
2293538	Abidjan	297.90564	24.76	1958

5. Machine Learning Models



Perimeter of the Analysis

Worked directly on the NetCDF files

World average temperature

World average precipitation

Historic Data

Forecast data (RCP 8.5 & RCP 4.5)

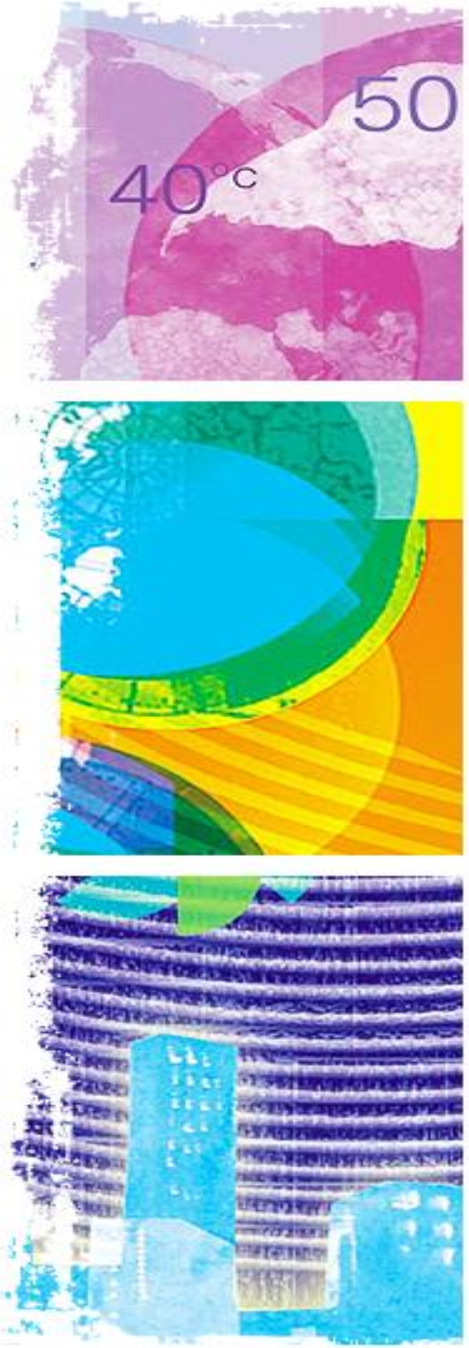
1950

2018

2100

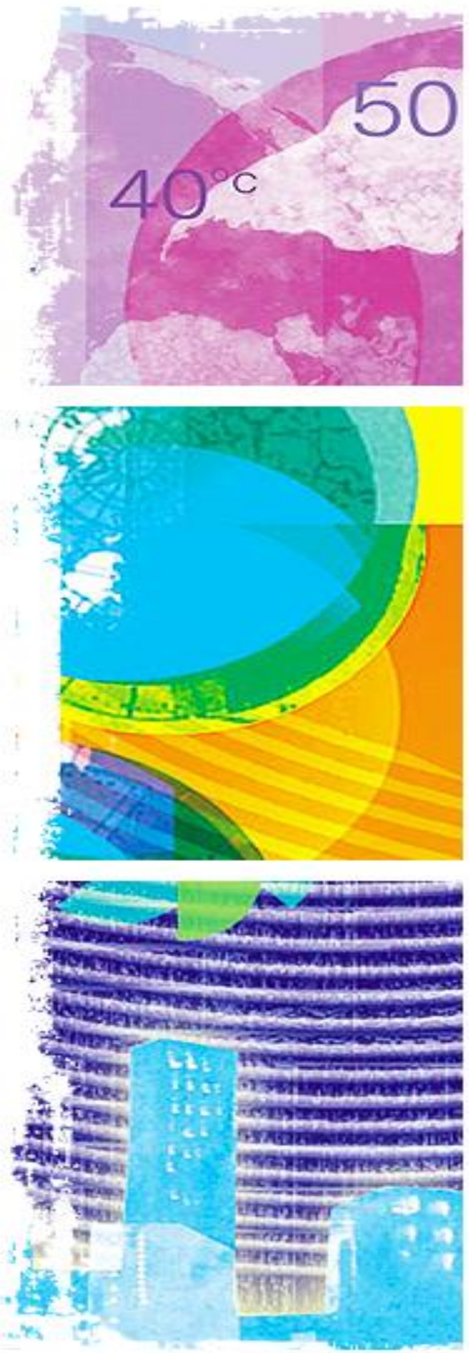
Time Series Analysis

Demo prediction to 2050



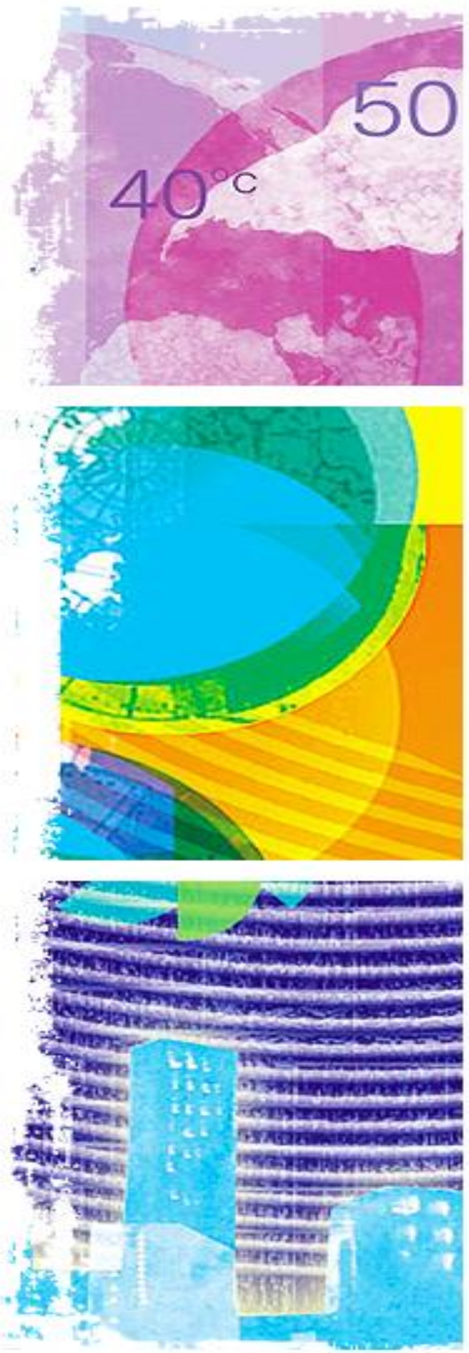
Process

- Data preparation
- Univariate time series analysis : MA, ARIMA & SARIMA
- Multivariate time series analysis : VAR & SARIMAX
- Demo with the best performing model



Data Preparation

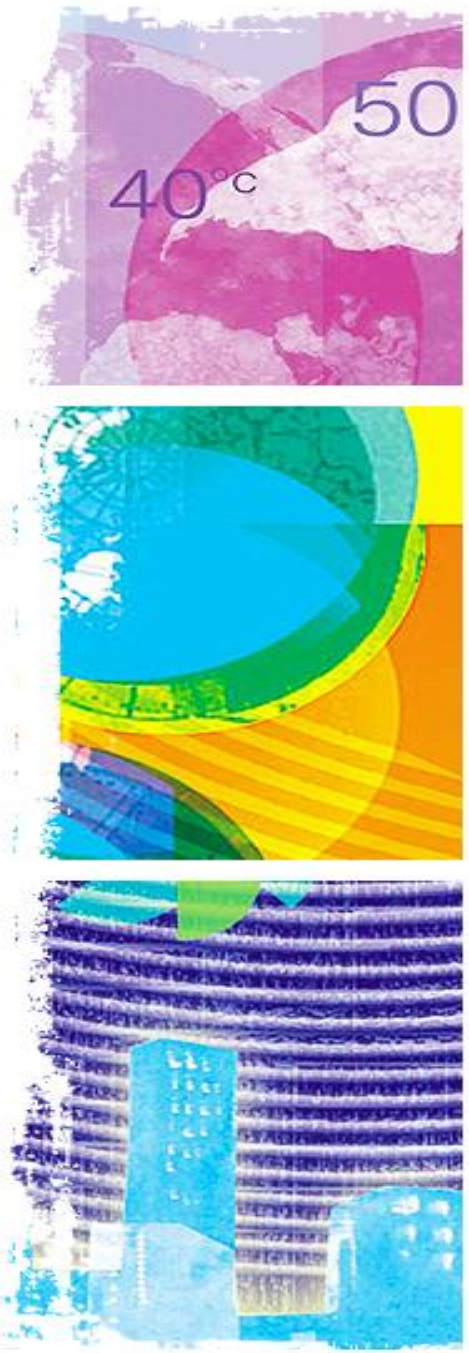
```python
# Load the netCDF file for Mean Temperature
d1 = xr.open_dataset('Worst_senario rcp8.5/dataset-sis-biodiversity/BIO01_ipsl-cm5a-lr_rcp

# Extract the annual mean temperature data
annual_mean_temp = d1['BIO01'].mean(dim=['latitude', 'longitude'])

# Convert the data to a pandas DataFrame
temp = annual_mean_temp.to_dataframe(name='temperature')

# Reset the index to make 'time' a column
temp = temp.reset_index()

# Convert the 'time' column to a pandas datetime object
temp['time'] = pd.to_datetime(temp['time'])

# Set the 'time' column as the index
temp = temp.set_index('time')

# Split the data into training and test sets
temp_train = temp.loc['1950-01-01':'2000-01-01', 'temperature']
temp_test = temp.loc['2001-01-01':'2018-01-01', 'temperature']
```



Univariate time series model comparison

	Temperatures	Precipitation
Moving Average model (MA)	- **RMSE**: 0.21 - **R2**: 0.21253 - **MAPE**: 0.0611975 - **AIC**: -39.4627950	- **RMSE**: 0.00 - **R2**: 0.31253 - **MAPE**: 0.861147645 - **AIC**: -394.94749450
Autoregressive integrated moving average model (ARIMA)	- **RMSE**: 0.24 - **R2**: 0.27838662 - **MAPE**: 0.06765037 - **AIC**: -14.4371690	- **RMSE**: 0.00 - **R2**: 0.421592446 - **MAPE**: 1.0961944 - **AIC**: -1058.2390
Seasonal ARIMA model (SARIMA)	- **RMSE**: 0.629466873 - **R2**: 0.00629996268 - **MAPE**: 0.199555 - **AIC**: -50.2901	- **RMSE**: 3.823029e-10 - **R2**: 0.12914789 - **MAPE**: 1.056691480 - **AIC**: -50.29013

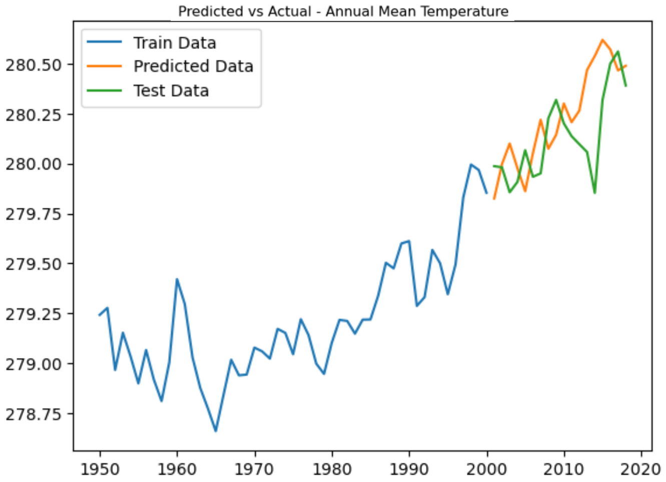
Univariate Time Series Model Comparison

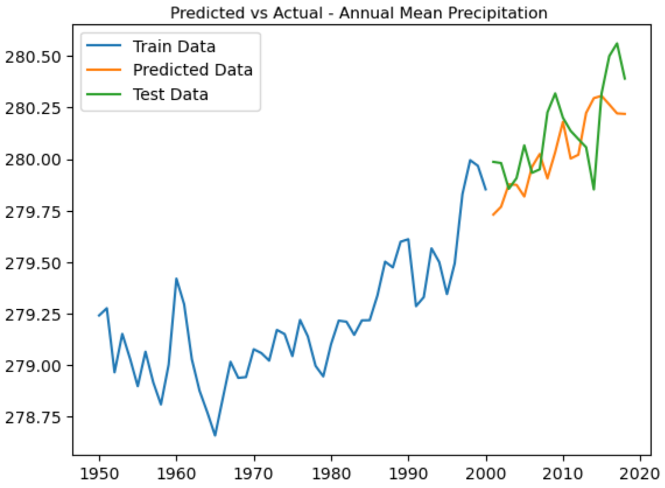


Temperature prediction

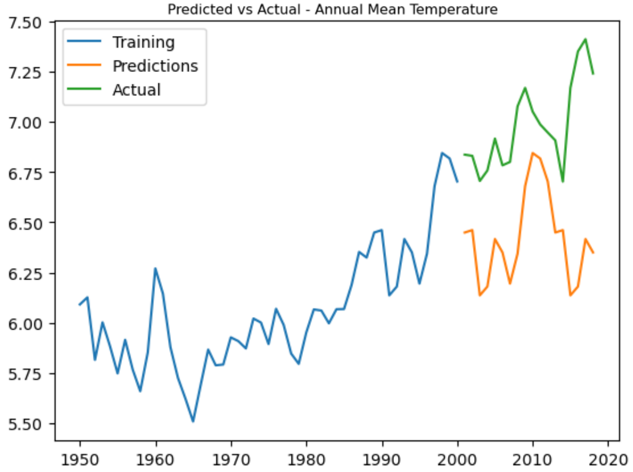
Precipitation prediction

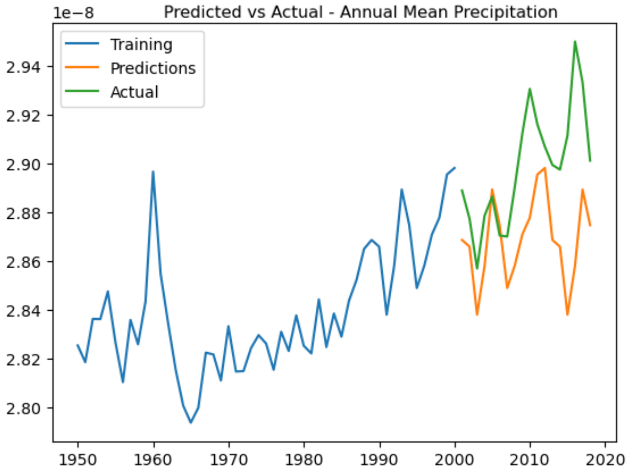
Autoregressive integrated moving average model (ARIMA)

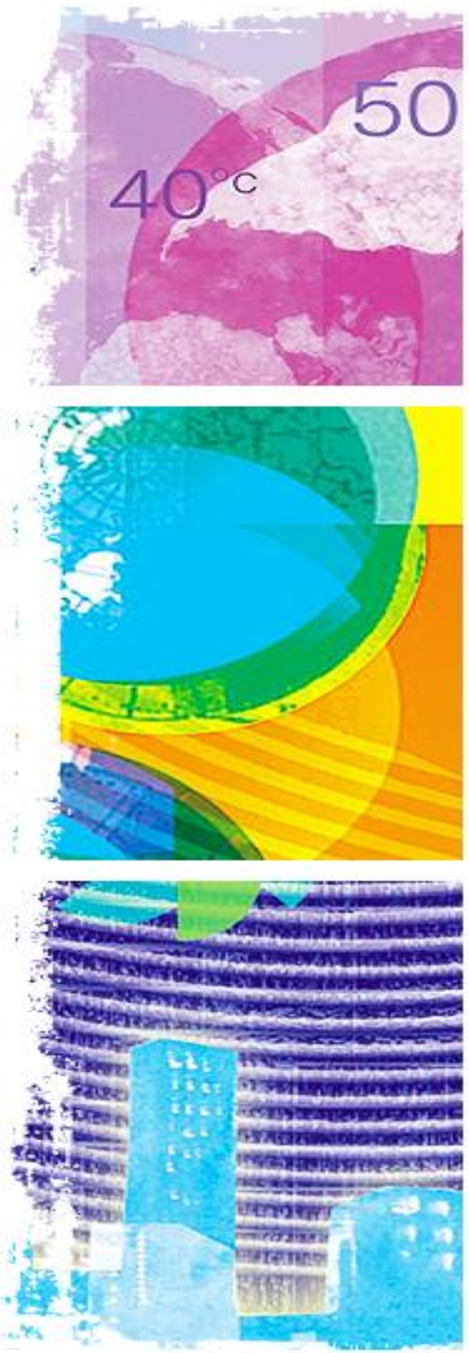




Seasonal ARIMA model (SARIMA)

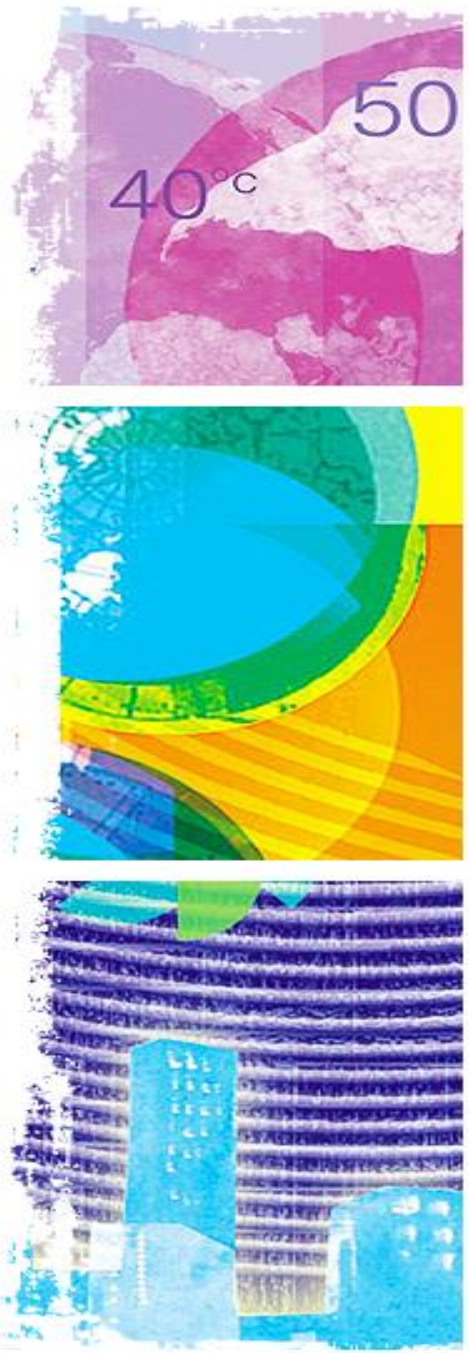






Multivariate time series model comparison

	Temperatures	Precipitation
vector autoregressive model (VAR)	- **RMSE**: 0.35914250 - **R2**: 0.1320513 - **MAPE**: 432.89421 - **AIC**: 326.47307	- **RMSE**: 4.1649868e-10 - **R2**: 0.1140740 - **MAPE**: 326.47307 - **AIC**: 326.47307
SARIMA model with exogenous factor (SARIMAX)	- **RMSE**: 0.157808 - **R2**: 0.5395002 - **MAPE**: 0.04484432 - **AIC**: -3.893976871	- **RMSE**: 2.0745384e-10 - **R2**: 0.52012773 - **MAPE**: 0.615481841 - **AIC**: -730.7987485

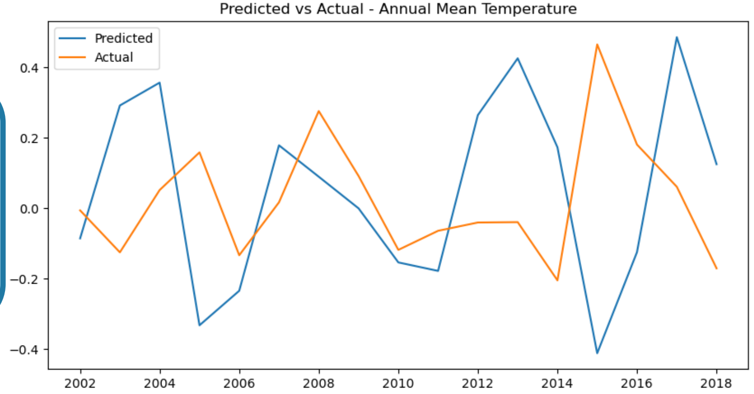


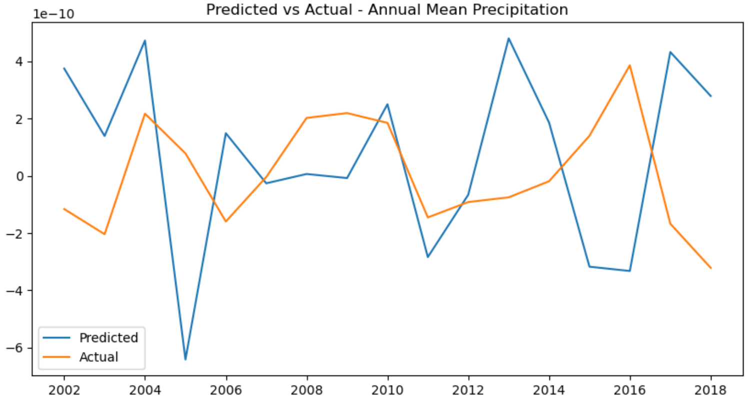
Multivariate Time Series Model Comparison

Temperature prediction

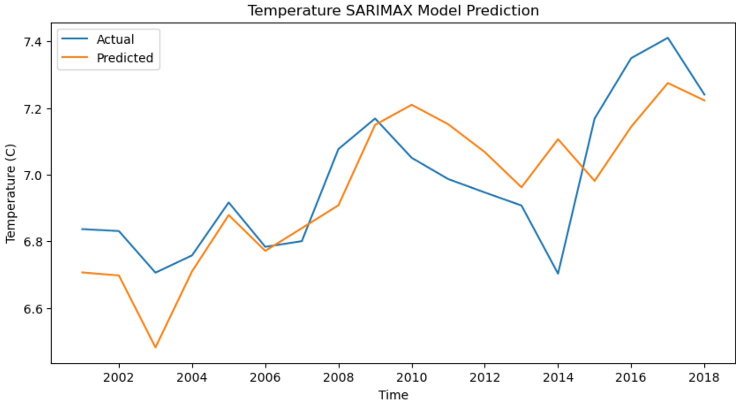
Precipitation prediction

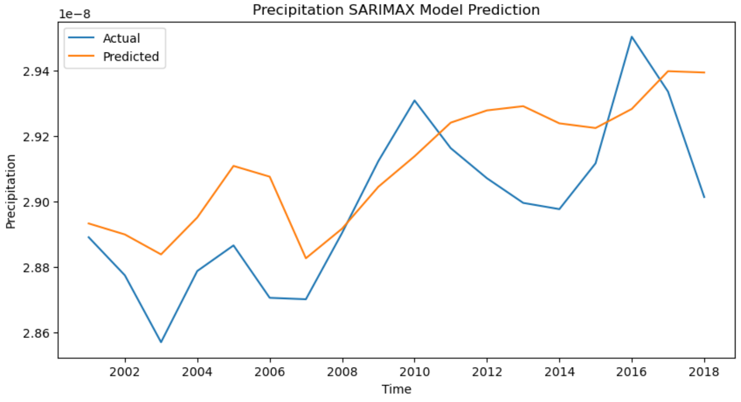
vector autoregressive model (VAR)



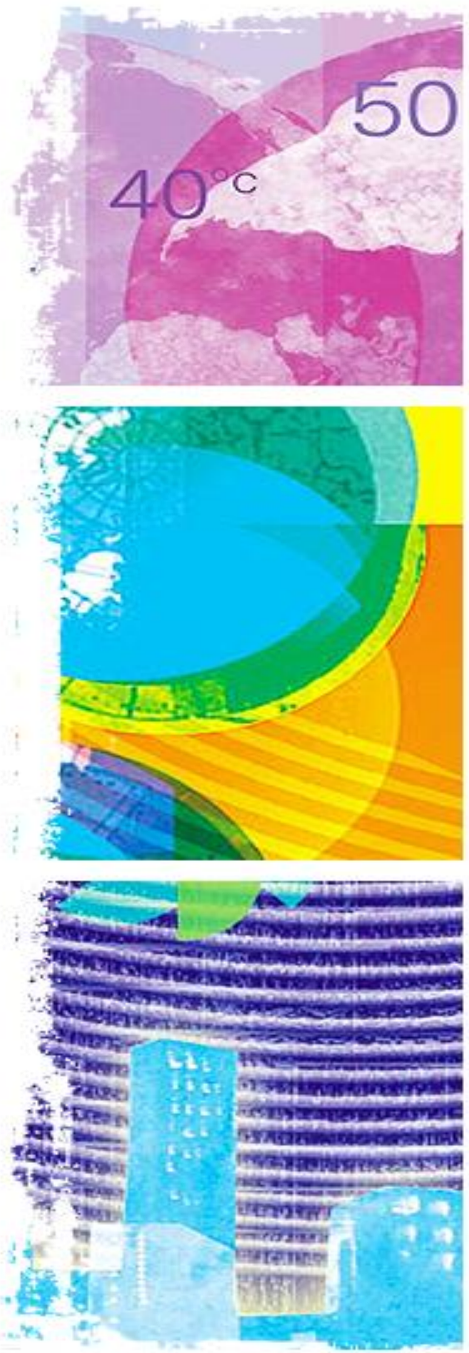


SARIMA model with exogenous factor (SARIMAX)

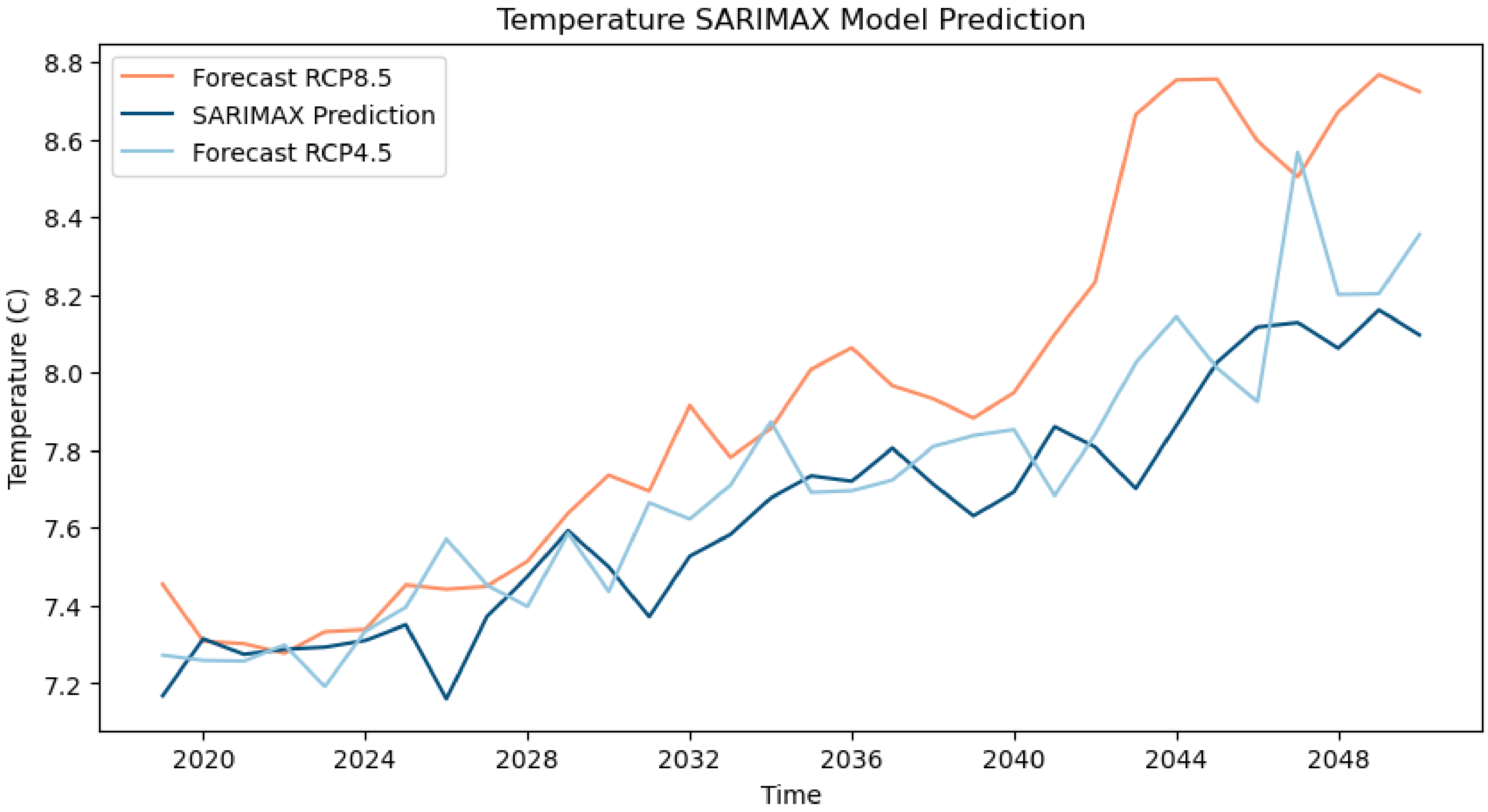




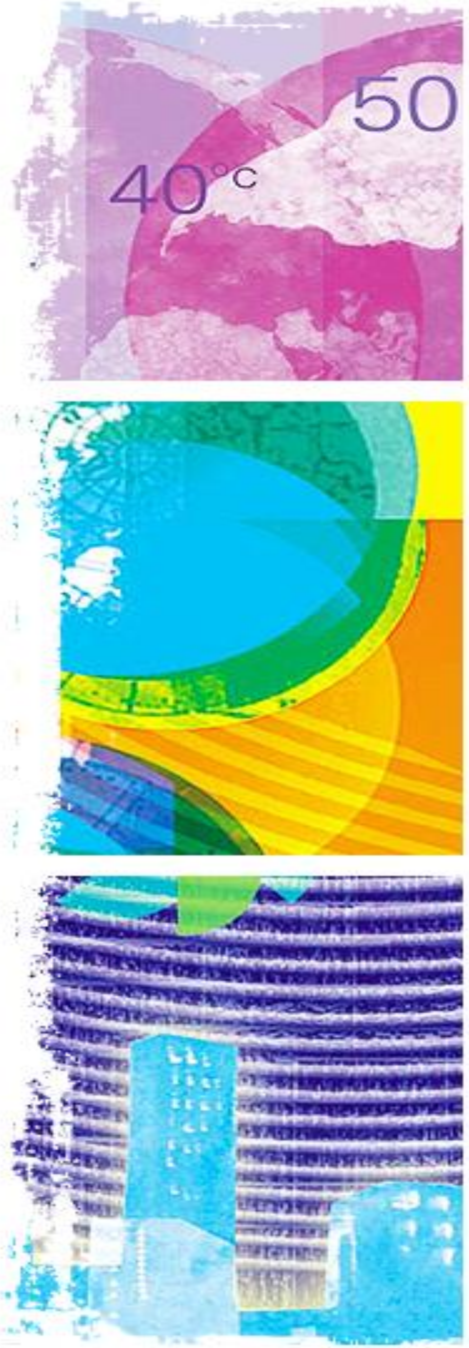
6. Demo & Insights

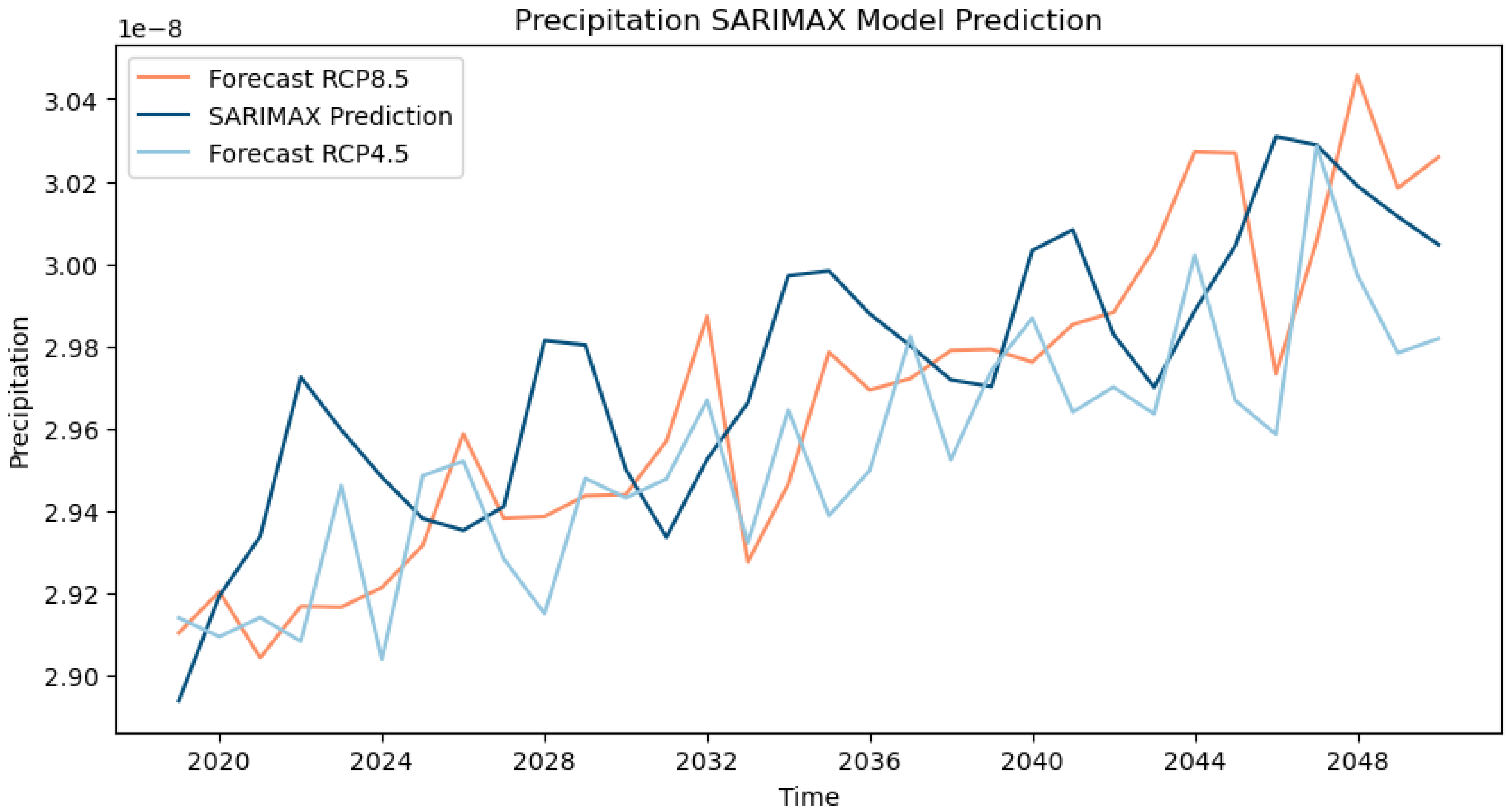


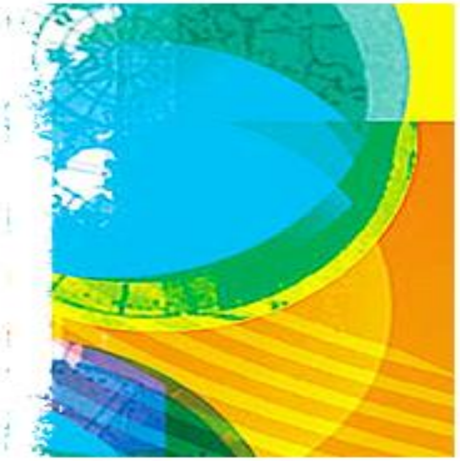
Demo : 2050 prediction using SARIMAX

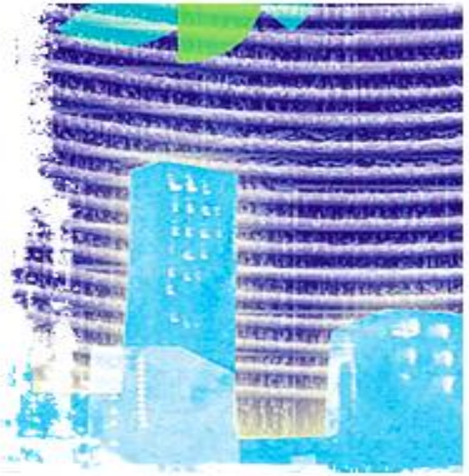


Demo : 2050 prediction using SARIMAX

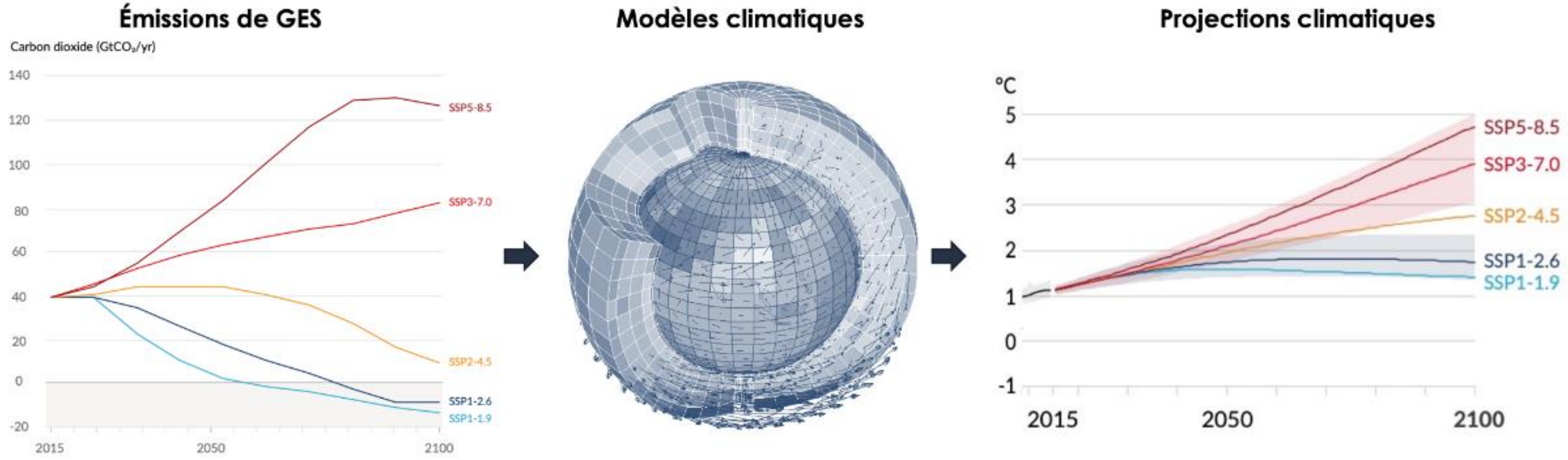








Real Climate Forecasting

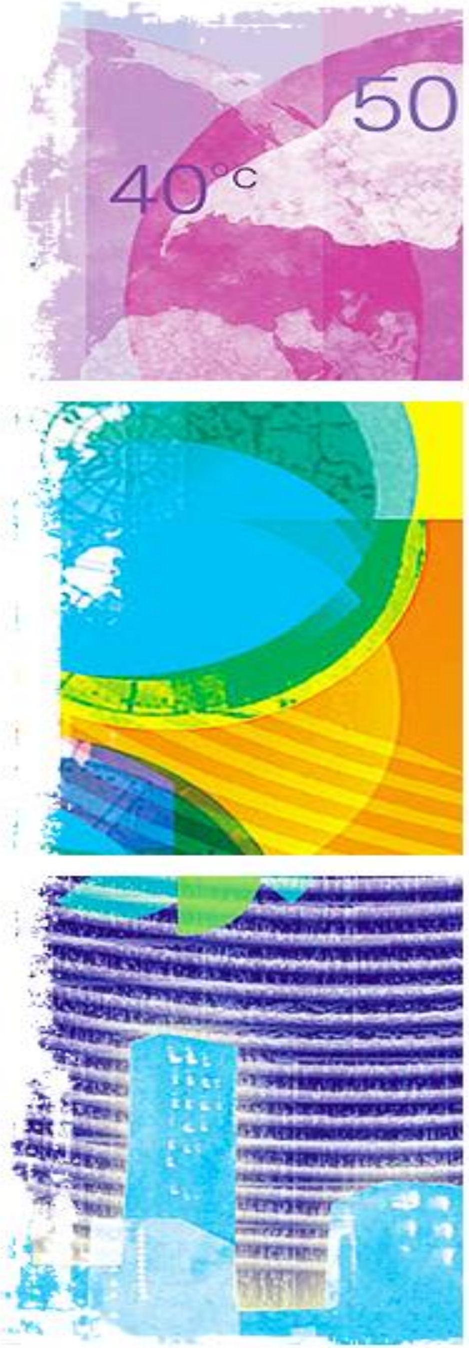


Carbone 4 : Explanation of a General Circulation Model

Conclusion

How is our cities' climate going to change in the future ?

Highlights



Highlights

Challenges:

- Handling gridded data
- Managing expectations

Learnings:

- Tool : Spyder
- Data format & extraction : NC files & NetCDF4 library
- Working in iterations

Improvements:

- With more time : check the forecast for specific cities and compare them to the official forecast
- Used a daily time series data


60 °C
50 °C
40 °C
35 °C
Thank You !