Data Analysis

Climate Liveability Index

Kaci BOURGUA

MARCH 2023

Table of content

Contents

1. Introduction & Use case

Given the increasing concerns about the impact of global warming on the climate and environment, it has become more important than ever to realize the impact it will have on our lives.

One way of doing so it to investigate the liveability of our cities, regarding climate metrics. Knowing which cities have a desirable climate for living will become increasingly important, with the rising global temperatures. We already have climate refugees, and in the future, we can imagine a trend of expatriation to cities with a more moderate climate to escape extreme heat or cold.

The idea behind this analysis was to identify cities with a desirable climate for living in the future, and the cities where it would be less pleasant to live, on a climate perspective with forecast data to 2100.

The focus of this analysis is on climate-related variables that reflect heat and humidity. I have used those variables to create a climate liveability index to identify cities with a desirable climate for living in the future.

This analysis answers the question: how liveable our cities will be in the future regarding heat and humidity?

The climate liveability index was used as the primary measure to determine the climate desirability of each city. With this dataset, we can explore the relationships between various climate-related variables and the climate liveability index to identify cities with undesirable climate for living in the future.

There are various use cases for this analysis:
On a business perspective, real estate developers and investors looking to invest in properties can consider the climate desirability as a factor in their investment decisions. But more importantly, a whole spectrum of companies working to innovate and make the world more liveable would be interested to know which cities to target for air quality devices or heat mitigation products.

On a public policy perspective, it is vital for policymakers to anticipate and develop resilience strategies and implement climate adaptation measures to improve the liveability of their cities.

For instance, cities with higher mean temperature in the warmest quarter might need to prioritize tree planting, green roofs, or other measures to reduce the heat island effect and provide shade for residents during the hottest months.

My plan for this analysis:

1. Obtain Data from Copernicus
2. Extract the data for a specific list of cities and obtain CSV files for each indicator
3. Clean CSVs file to dataframes
4. Export clean data to MySQL for processing
5. Join datasets into one
6. Process data to use appropriate metrics
7. Decide on appropriate Time Series model (or other) for the prediction of one variable (like mean temperature)
8. Feature Selection and Engineering
9. Predict the variable based on appropriate features and comparing the predictions to the forecast of the climate model

2. Data and data sources

For this project, I have used two data sources. My first source of data is The GeoNames geographical database, which covers all countries and contains over eleven million placenames that are available for download free of charge. From this source, I have gathered a dataset of all the cities around the world with more than 1000 inhabitants, with their city name, country name, population and latitude/longitude.
I have used the geographic information of this dataset to extract the data from the second dataset.

My second source of data is Copernicus, the European Union's Earth Observation Program, which provides a wide range of data on various aspects of the Earth's environment. The program is made up of a series of satellite missions and ground-based monitoring systems, which gather data on topics such as climate change, land use, oceanography, and atmospheric composition.

Copernicus provides a range of data products that are useful for The Intergovernmental Panel on Climate Change (IPCC) in their reports, particularly in assessing the impacts of climate change on the environment.

I wanted to work on this data because it is the most reliable source for environmental data (from satellites), and it is used in all the majors reports that are warning the public about global warming.

From the dataset "**Global bioclimatic indicators from 1950 to 2100 derived from climate projection**", I have downloaded 6 indicators that were the most relevant to build my climate liveability index:

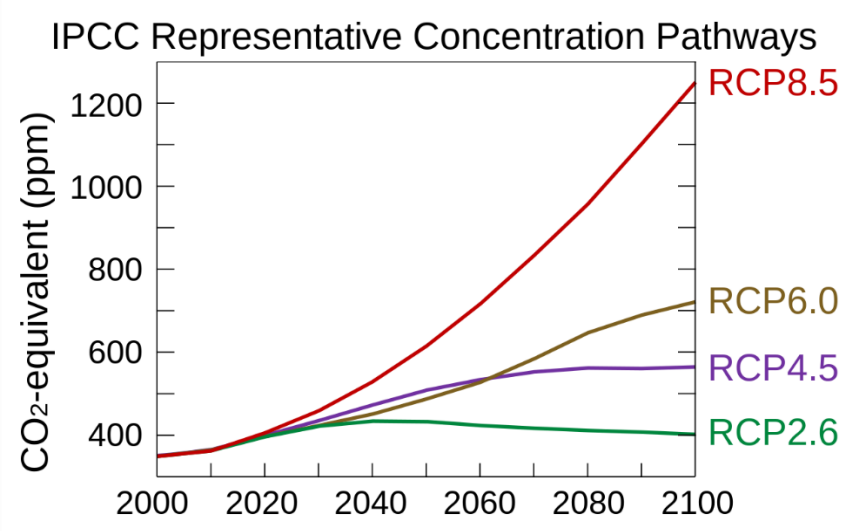
Variable Name	Unit	Description
Annual mean temperature (BIO01)	K (Converted to °C)	Annual mean of the daily mean temperature at 2 m above the surface. This indicator corresponds to the official BIOCLIM variable BIO01 that is used in ecological niche modelling.
Mean temperature of warmest quarter (BIO10)	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.

This is a gridded dataset (matrix with 4 dimensions: latitude, longitude, time, indicator value), that comes in a NetCDF-4 format (NC file).

These are yearly values from 1950 to 2100, with the historical data stopping at 2018 and the forecast data starting at 2019.

The forecast data follows the RCP8.5 scenario (Representative Concentration Pathways), which is the worst one in the IPCC framework (it was replaced by the Shared Socioeconomic Pathways). So this data reflects the least optimistic climate forecast.



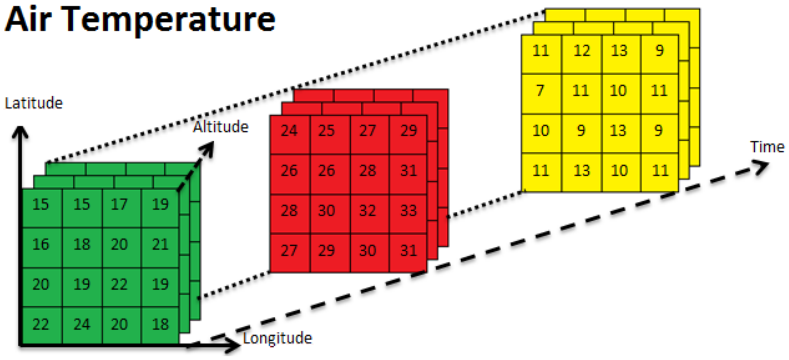
Source 1 : All forcing agents' atmospheric CO2-equivalent concentrations (ppm)[1] from the IPCC AR5 report

3. Data collection & extraction

Copernicus offers two methods to collect data from a dataset. You can either use an API or just submit a request form. Either way, you have to select specific fields like the variables, the model used for the prediction, the climate scenario (RCP) and other parameters.

Once all the fields have been selected, you can either copy paste the API code generated or just submit the request, wait a few minutes for it to be validates, and download your data. I have used the latter for simplicity (and because the files were quite large).

Once the data is collected, the next step is to extract it, as it is stored in an NC file, which is a format for storing multidimensional scientific data (variables) such as humidity, pressure or temperature as seen bellow:



Source 2: geoserver.geo-solutions

Before extracting the data, I first had to check if there was a distortion of the latitude and longitude (projected grid, rotated grid, or curvilinear grid) that would require extra treatment. Here it was not the case, the latitude and longitude lines were constant.

To access the data, I have used the tool Spyder, to have a better view of my data, as I was working in iteration. Indeed, I first tried to extract the data of one variable, for one single location (latitude & longitude), before adapting my script for multiple locations, trying it on a few cities and then only, applying it to all the cities and variables.

For this I created a Python script to extract a variable data using the latitude and longitude of a city (from a list of the top 100 cities in population size).

```
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)
```

This part creates the structure of the dataframe where the data will be stored.

```
# 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")
```

Then I had to loop through a list of cities and get their latitude and longitude, and take the closest latitude and longitude points from the gridded data to get the variable value and store it in the dataframe before saving it as a CSV file.

Since this loop goes through a list of 100 cities, it returns 100 different CSV files, for each indicator. Those files needed to be merged, and this is what the last part of the code bellow does (merge all the files from a folder).

```
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)
```

4. Data cleaning and Exploratory data analysis

- ## Cleaning

I first started cleaning the cities CSV file from which I have based my data extraction on. Starting with the column names:

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

I then split the latitude and longitude which were in a single column:

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

I then dropped columns that contained too much missing values or were not relevant for my analysis:

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

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

I then start cleaning the data I have retrieved from my NC files, starting with converting the temperatures from Kelvin to Celsius:

```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
```

I then split the date format to only keep the year (since the months do not change as it is a yearly value):

```python
#Cleaning the date format to only keep the year
final_df6['year']=final_df6['year'].str.split('-').str[0]
```

Python

The data was pretty reliable given its source, so there was no missing value or inconsistencies to handle.

- **Index**

I created my index using the 6 bioclimatic indicators that I had. I first scaled my data and then multiplied their values by the weight I had assigned to each indicator. The result is an index between 0 and 1, the higher the index, the less desirable the climate conditions of the city are.

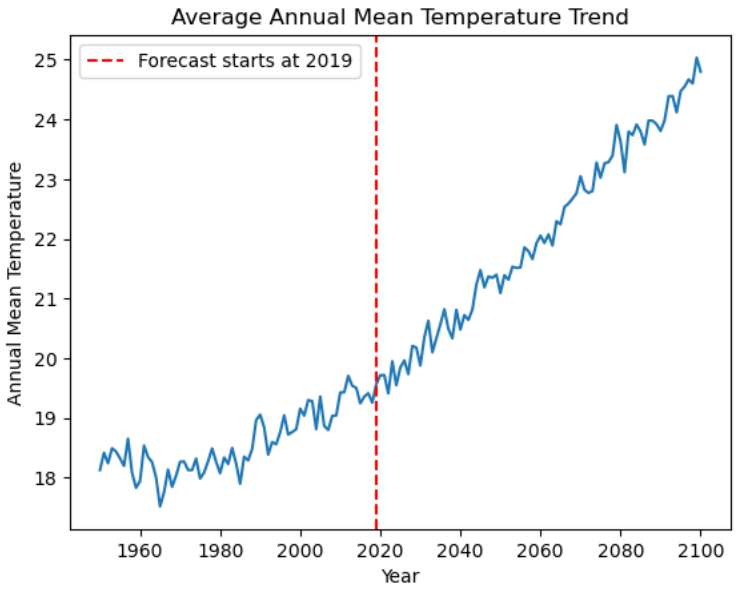
This index basically shows that the more humid and warm a city is, the less desirable its climate will be. It is a subjective approach to climate desirability of course but can be a good indicator of how the heat/humidity indicators evolve in time.

Here is the code that I have used to create my index:

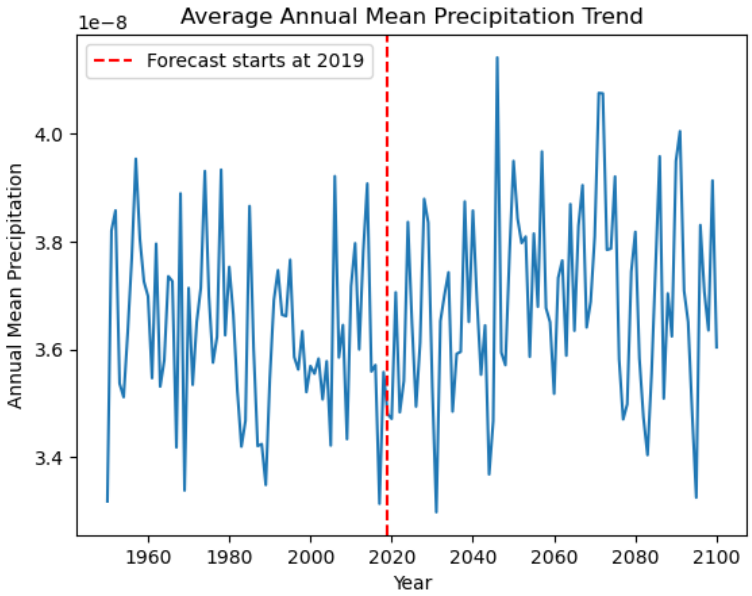
```python
from sklearn.preprocessing import MinMaxScaler

# 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
```

✓ 0.3s Python

- **Exploratory data analysis**

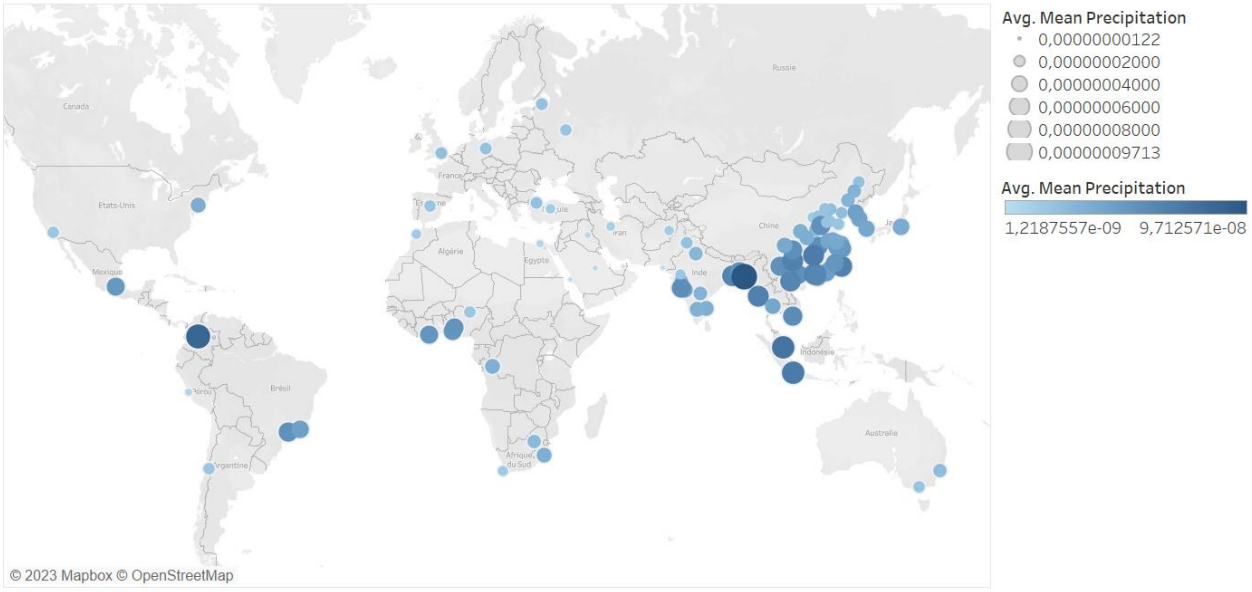


Since 1950, we can clearly see an increase of the average temperatures in this sample of 100 cities. The however clearly shows a steep increase in the next 80 years. This reflects the worst scenario in terms of CO2 emissions.



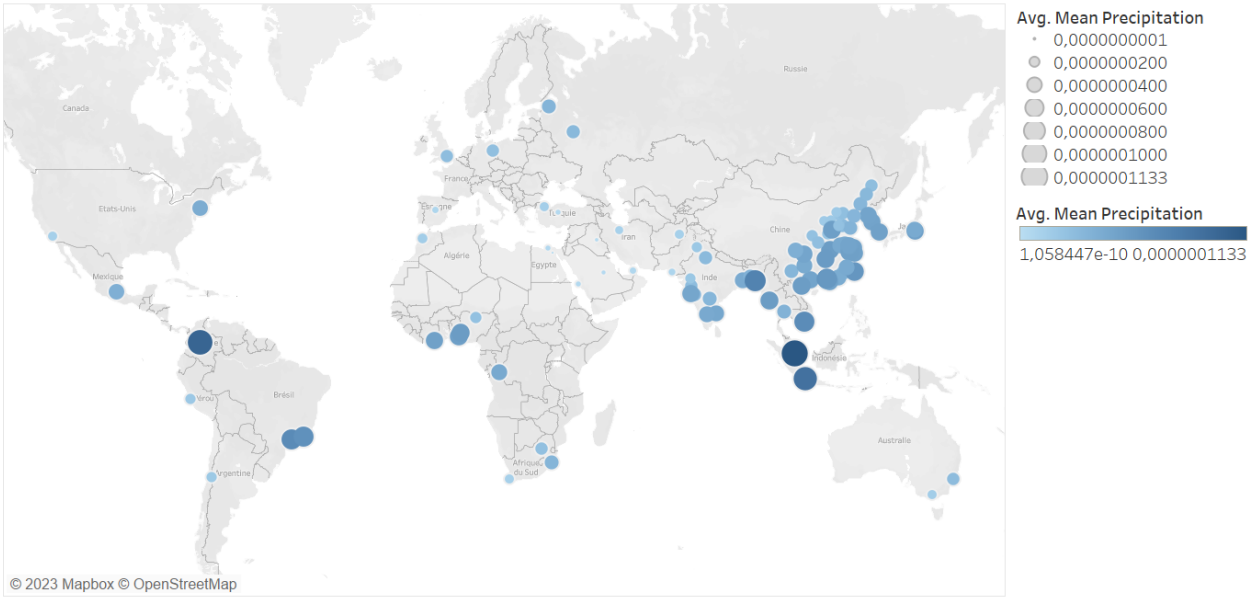
For the precipitation, it is more difficult to identify a trend, as precipitation seem to follow a seasonal pattern. However, we can see that there are more extreme values predicted (years with high precipitation, and others with very low precipitation).
We can therefor expect more droughts and floodings according to this forecast.

Mean precipitation in 2018



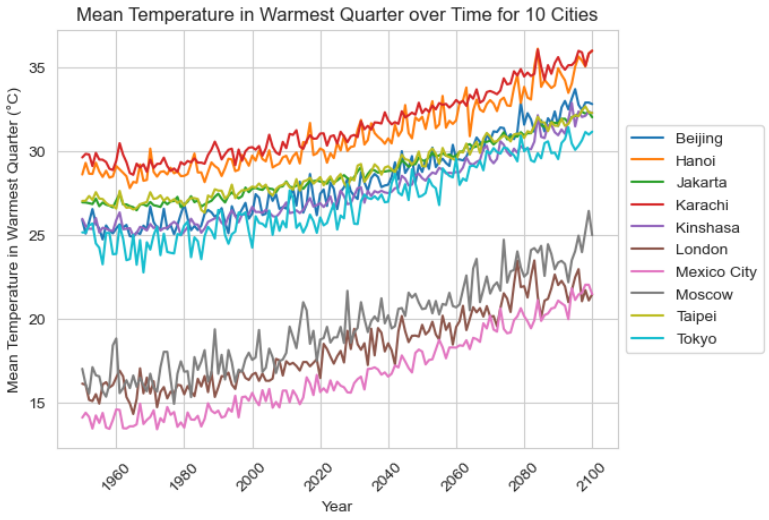
Map based on Longitude (generated) and Latitude (generated). Color shows average of Mean Precipitation. Size shows average of Mean Precipitation. Details are shown for Name and Country Name En. The data is filtered on Year, which ranges from 2018 to 2018.

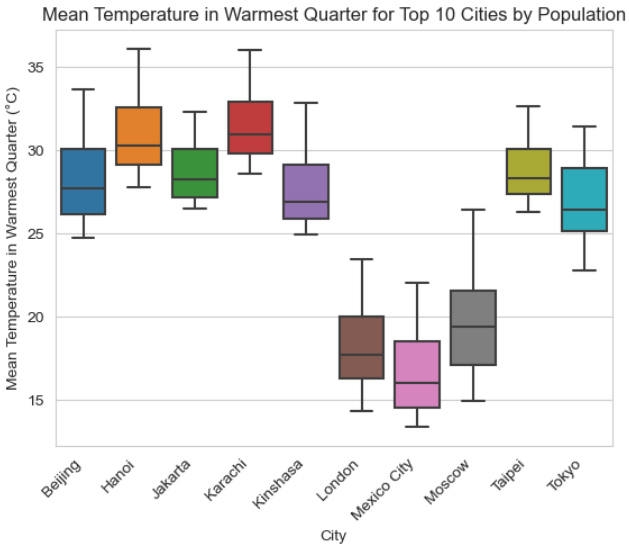
Mean precipitation in 2100



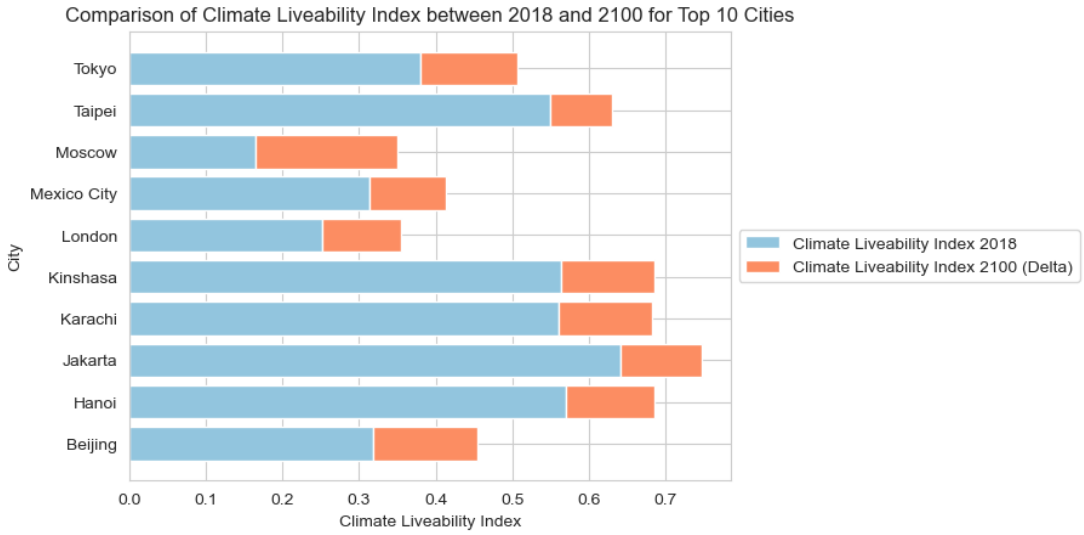
Map based on Longitude (generated) and Latitude (generated). Color shows average of Mean Precipitation. Size shows average of Mean Precipitation. Details are shown for Name and Country Name En. The data is filtered on Year, which ranges from 2100 to 2100.

Looking into the precipitation data, we can see that in 2100, the cities around the Mediterranean see will receive less rain on average, compared to 2018.



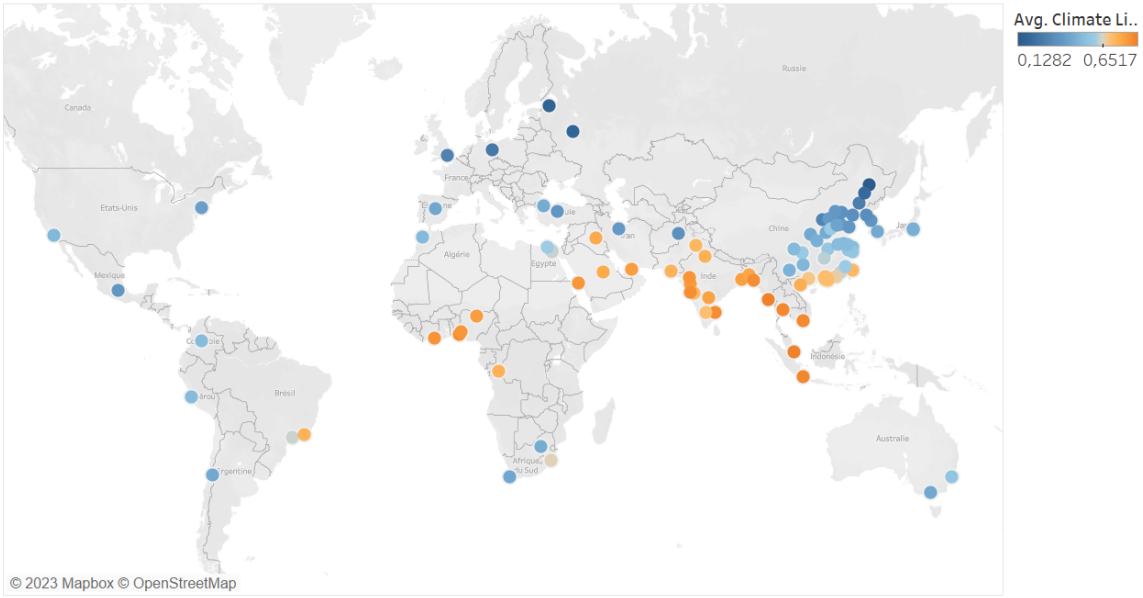


If we focus on the extreme heat episodes, we see that historically, the temperatures of the warmest quarter of the year are overall constant since 1950, but in the forecast, we see an increase of those temperatures. This means that there will be an increasing amount of heat waves during the warmest quarter of the year. Moscow will experience the most variability in its temperatures.



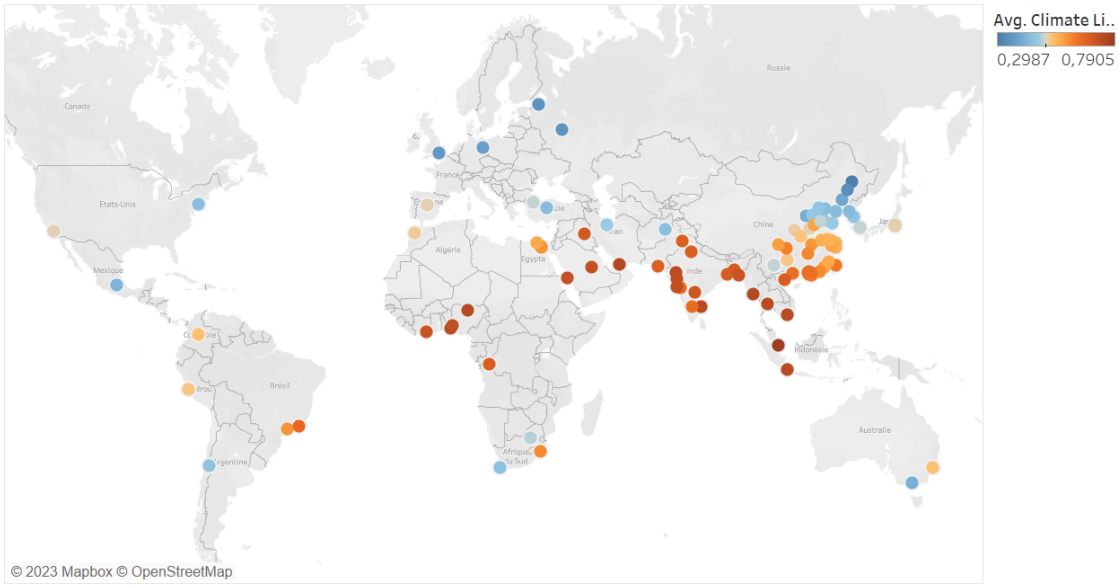
Looking at our Climate Liveability Index, we can see that for a sample of 10 cities around the world, this index will deteriorate between 2019 and 2100. This means that those cities will be hotter and/or more humid. Moscow seems to be the city that will have its climate the most impacted in the next 80 years according to the forecast.

Liveability Index for 2018



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



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.

The liveability index map shows us how the most inhabited cities around the world will be impacted by climate change by 2100. We clearly see that cities around the southern hemisphere will be unequally impacted, in particular cities of the African continent, the Middle-East and the Southern part of the Asian continent.

5. Data base type selection

The next step was creating my database. I decided to use SQL because this type of database offers a lot of possibilities to run complex queries fast, and it allows me to manipulate all my tables and join them to create my final document for my analysis.

Why SQL and no NoSQL? 5 main reasons:
- SQL databases are relational; NoSQL databases are non-relational. - SQL databases use structured query language and have a predefined schema. NoSQL databases have dynamic schemas for unstructured data.
- SQL databases are vertically scalable, while NoSQL databases are horizontally scalable.
- SQL databases are table-based, while NoSQL databases are document, key-value, graph, or wide-column stores.
- SQL databases are better for multi-row transactions, while NoSQL is better for unstructured data like documents or JSON.
This allows me to quickly make queries and join different information from various tables.

Another advantage of SQL is its integration to Python. For this project, I have used the library “sqlAlchemy” to create the tables of my database directly from the Pandas dataframes of each indicator. It was very easy and faster than creating them directing on MySQL.

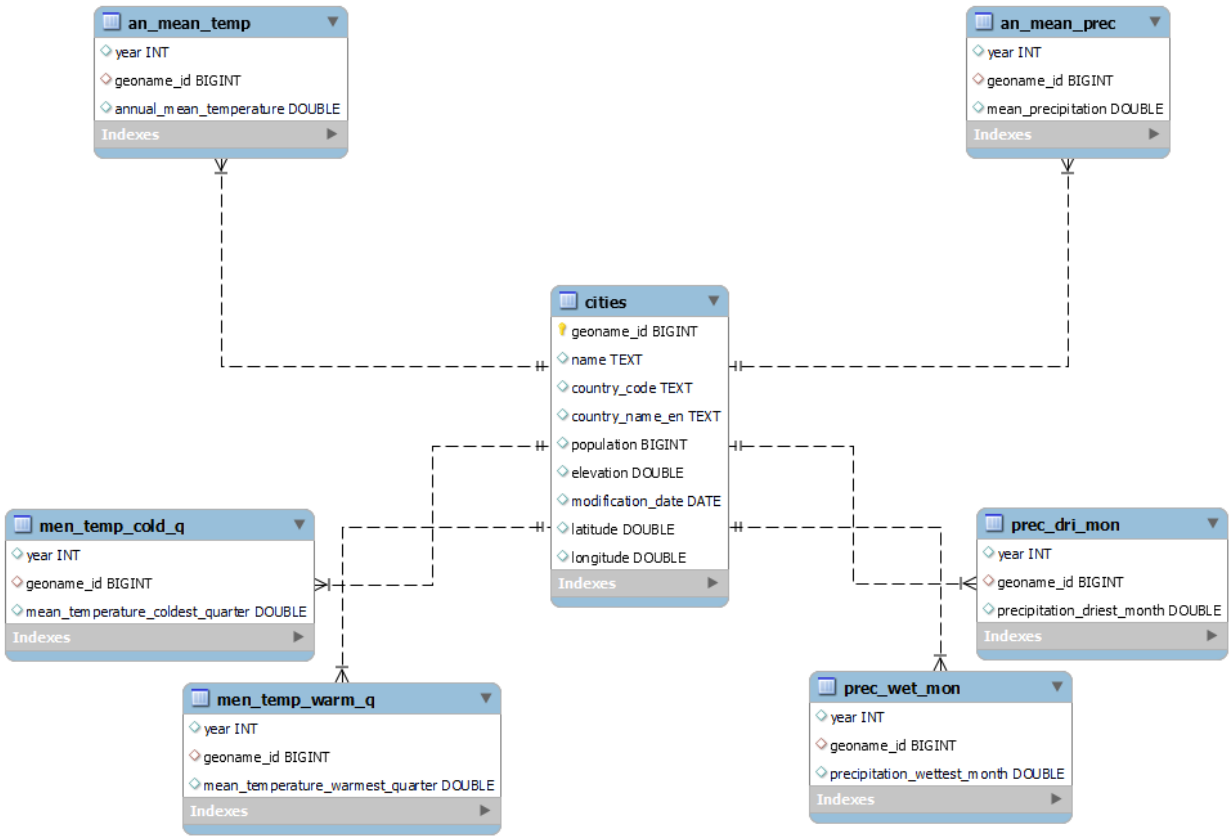
Once my database created, I then switched to MySQL Workbench to manipulate my tables and create the entity relationship model of this database, to clarify the different entity tables I will use and specify their relationships.

6. Database Creation & ERD

As previously stated, 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)
```

The diagram for the entity relationship model of my database is as follow :



The cities table is at the center and contains all the information about each city (name, country, population, latitude & longitude…). Then there is a table for each indicator, connected to the cities table by the geoname_id of the city (primary key of the cities table).
This diagram reflects the way my data was extracted.

7. SQL Queries

Here are my 5 SQL queries from MySQL:

```sql
-- Query 1 : Cities grouped by and ordered by countries
select country_name_en, name
from cities
group by country_name_en, name
order by country_name_en, name asc;
```

country_name_en	name
Afghanistan	'Alāqahdārī Atghar
Afghanistan	'Alāqahdārī Dīshū
Afghanistan	'Alāqahdārī Gēlān
Afghanistan	'Alāqahdārī Sarōbī
Afghanistan	'Alāqahdārī Shāh Jōy
Afghanistan	'Alāqahdārī Yōsuf Khēl
Afghanistan	'Alāqahdārī-ye Almār
Afghanistan	'Alī Khēl

```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 3 : Top 10 cities in France, by population
select name, population, country_name_en
from cities
WHERE country_name_en = 'France'
order by population desc
limit 10;
```

name	population	country_name_en
Paris	2138551	France
Marseille	870731	France
Lyon	522969	France
Toulouse	493465	France
Nice	338620	France
Nantes	318808	France
Strasbourg	274845	France
Bordeaux	260958	France
Montpellier	248252	France

```
-- 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

```
-- Query 5 : Join all the tables (& export in csv for the analysis)
SELECT amt.geoname_id,
c.name,
c.country_name_en,
c.population,
amt.year,
amt.annual_mean_temperature,
amp.mean_precipitation,
mtcq.mean_temperature_coldest_quarter,
mtwq.mean_temperature_warmest_quarter,
pdm.precipitation_driest_month,
pwm.precipitation_wettest_month
from an_mean_temp amt
left join an_mean_prec amp
on amt.geoname_id=amp.geoname_id and amt.year=amp.year
left join men_temp_cold_q mtcq
on amp.geoname_id=mtcq.geoname_id and amp.year=mtcq.year
left join men_temp_warm_q mtwq
on mtcq.geoname_id=mtwq.geoname_id and mtcq.year=mtwq.year
left join prec_dri_mon pdm
on mtwq.geoname_id=pdm.geoname_id and mtwq.year=pdm.year
left join prec_wet_mon pwm
```

```
on pdm.geoname_id=pwm.geoname_id and pdm.year=pwm.year
left join cities c
on pwm.geoname_id=c.geoname_id
order by name asc, year asc;
```

geoname_id	name	country_nam	populatic	year	annual_mean_te	mean_precipitation	mean_temperature	mean_temperature_	precipitation_drie	precipitation_wettest_
2293538	Abidjan	Côte d'Ivoire	6321...	1950	298.82852	0.000000047875186	297.37543	299.75824	0.0000000128...	0.0000001292456
2293538	Abidjan	Côte d'Ivoire	6321...	1951	298.45282	0.000000044937313	296.61664	299.76868	0.0000000147...	0.00000013062728
2293538	Abidjan	Côte d'Ivoire	6321...	1952	298.19434	0.000000043604555	296.38623	299.60165	0.0000000216...	0.00000010903607
2293538	Abidjan	Côte d'Ivoire	6321...	1953	298.4637	0.000000049382276	296.84213	299.4366	0.0000000205...	0.00000012645171
2293538	Abidjan	Côte d'Ivoire	6321...	1954	298.43268	0.000000042335433	296.18115	299.8955	0.0000000129...	0.00000009929957
2293538	Abidjan	Côte d'Ivoire	6321...	1955	298.13373	0.0000000483454	296.44736	299.26602	0.0000000127...	0.00000012258177
2293538	Abidjan	Côte d'Ivoire	6321...	1956	298.47598	0.000000047298784	296.6787	299.6376	0.0000000127...	0.00000012721286
2293538	Abidjan	Côte d'Ivoire	6321...	1957	298.37076	0.000000048840015	296.5696	299.5075	0.0000000128...	0.00000012578295
2293538	Abidjan	Côte d'Ivoire	6321...	1958	297.90564	0.00000004709963	296.23068	299.31342	0.0000000127...	0.00000014131797

Result 5

8. Conclusion

The data from the CMIP5 climate projections for the RCP8.5 scenario (the most pessimistic in terms of CO2 emissions) show that on average, our world will be warmer, and it will impact the climate of our cities differently.

Using a sample of the 100 most populated cities in the world, we can already see where climate adaptation measures will be most needed, and what cities will be ideal in terms of combined effects of warm temperatures and humidity.

The cities in the norther part of Europe seem to be less impacted than the other regions in terms of heat and are therefore good candidates for climate expatriation.

If you happen to dislike living in a hot and humid climate, you might want to avoid South-Asia, South-East Asia and most of the African continent around 2100.

Link to the GitHub repository of this project:
https://github.com/Kaci213/Bootcamp/tree/master/Project_Final