Supervising teacher: Grigoris Tsagkatakis

University of Crete- Computer Science Department   
2024



Visualization of Multidimensional Environmental Data

Boumpouraki Athanasia

Table of Contents

[1 Introduction 3](#_Toc171004613)

[1.1 Background Information 3](#_Toc171004614)

[1.2 Objectives 3](#_Toc171004615)

[1.3 Scope 3](#_Toc171004616)

[2 Literature Review 4](#_Toc171004617)

[2.1 Previous Work on Environmental Data Visualization 4](#_Toc171004618)

[2.2 Key Methods and Tools Used in the Past 4](#_Toc171004619)

[2.3 Gaps in Current Research 4](#_Toc171004620)

[3 Methodology 5](#_Toc171004621)

[3.1 Data Collection 5](#_Toc171004622)

[3.1.1 Data Processing 5](#_Toc171004623)

[3.1.2 Visualization Tools and Techniques 6](#_Toc171004624)

[3.1.3 3D Model Creation and Data Projection 6](#_Toc171004625)

[3.2 Challenges and Solutions 6](#_Toc171004626)

[3.3 Summary of Key Resources and Methods 6](#_Toc171004627)

[4 Implementation 7](#_Toc171004628)

[4.1 Visualization Process 7](#_Toc171004629)

[4.1.1 Data Collection and Preparation 7](#_Toc171004630)

[4.1.2 Data Processing 8](#_Toc171004631)

[4.1.3 Visualization Creation 11](#_Toc171004632)

[4.1.4 Integration and Presentation 20](#_Toc171004633)

[4.2 Challenges 21](#_Toc171004634)

[5 Results 22](#_Toc171004635)

[5.1 Key Findings from Visualized Data 22](#_Toc171004636)

[5.2 Examples of Visualizations 22](#_Toc171004637)

[6 Interpretation 23](#_Toc171004638)

[7 Discussion 23](#_Toc171004639)

[7.1 Implications 23](#_Toc171004640)

[7.2 Comparison with Previous Research 24](#_Toc171004641)

[7.3 Applications 24](#_Toc171004642)

[8 Conclusion 25](#_Toc171004643)

[8.1 Summary 25](#_Toc171004644)

[8.2 Contributions 25](#_Toc171004645)

[8.3 Future Research 25](#_Toc171004646)

[9 References 25](#_Toc171004647)

# Introduction

Environmental data encompasses a broad spectrum of information pertaining to the natural world. This data includes key parameters such as temperature, precipitation, soil moisture, surface pressure, and wind. Interpreting raw environmental data presents several challenges due to its complexity and volume. Visualization addresses these challenges by transforming complex data into comprehensible visual representations, thereby enhancing understanding and facilitating informed decision-making.

## Background Information

Environmental data, including parameters such as temperature, precipitation, soil moisture, surface pressure, and wind, plays a critical role in various fields. In climate science, understanding temperature and precipitation trends is vital for studying climate change and predicting future scenarios. This data aids in identifying long-term trends, assessing the impact of human activities on the climate, and developing predictive models. For agriculture, soil moisture and temperature data are essential for optimizing irrigation and crop selection, thereby enhancing productivity and sustainability. Moreover, environmental data is indispensable for policymakers, helping in the formulation of regulations related to environmental protection, disaster management, and sustainable development. Accurate environmental data assists in balancing economic growth with environmental sustainability.

The significance of environmental data extends across climate science, agriculture, and policy-making. However, the large volumes and multidimensional nature of this data make it difficult to analyse without effective visualization tools. Visualization transforms complex data into visual representations, making patterns, trends, and anomalies more apparent. These visualizations enhance understanding by making complex data accessible to scientists, policymakers, and the general public, supporting informed decision-making.

## Objectives

The primary objective of this thesis is to develop comprehensive visualizations of environmental data for the island of Crete. These visualizations aim to enhance the understanding of various environmental parameters and their temporal and spatial variations. The thesis seeks to demonstrate the power of visualization in uncovering hidden patterns in environmental data, supporting the development of more effective environmental policies, and fostering a deeper public understanding of environmental issues.

## Scope

The scope of this thesis includes creating static visualizations for individual years and parameters, providing detailed snapshots of environmental conditions. Additionally, the development of interactive maps will allow users to explore environmental data by selecting different parameters and years. The construction of 3D terrain models with overlaid environmental data will provide a more immersive and spatial understanding of the data. Furthermore, generating animations that show the temporal evolution of environmental parameters over multiple years will offer a dynamic view of environmental changes. The methodologies employed include data collection and preprocessing, advanced visualization techniques, and the development of interactive tools to facilitate user engagement with the data.

Through these visualizations, this thesis aims to fill existing gaps in the current research landscape, advancing the scientific understanding of environmental phenomena on Crete and providing a framework that can be applied to other regions and datasets.

# Literature Review

## Previous Work on Environmental Data Visualization

The visualization of environmental data has been an essential aspect of climate research, agricultural planning, and policy-making for decades. Numerous studies have explored various techniques to represent environmental parameters such as temperature, precipitation, soil moisture, and wind. Early works often relied on static maps and graphs to convey this information. Notable projects include the use of Geographic Information Systems (GIS) for mapping environmental data and remote sensing technologies to gather and visualize data over large geographic areas. The development of tools such as Google Earth Engine has revolutionized how researchers access and analyse environmental data, enabling large-scale data processing and visualization.

Several projects have focused on specific regions or phenomena, such as the visualization of climate change impacts on specific ecosystems or the mapping of drought conditions across agricultural lands. These projects have laid the groundwork for more advanced visualization techniques, highlighting the importance of clear and accessible data representation in understanding and mitigating environmental issues.

## Key Methods and Tools Used in the Past

A variety of tools and methodologies have been employed to process and visualize environmental data. Geographic Information Systems (GIS) remain a cornerstone for spatial data analysis and visualization, allowing for the integration of various data sources into comprehensive maps. Remote sensing software, such as ENVI and ERDAS IMAGINE, provides advanced capabilities for processing satellite imagery and extracting meaningful environmental information.

In recent years, the rise of open-source programming languages and libraries has democratized access to powerful visualization tools. Python has become a popular choice for data visualization due to its extensive library ecosystem. Libraries such as Folium, Matplotlib, and Plotly offer diverse functionalities for creating interactive maps, static plots, and complex visualizations. Folium, for instance, facilitates the creation of interactive maps that can display environmental data layers dynamically, enhancing user engagement and understanding.

Methodologies typically involve data collection from various sources, data preprocessing to clean and format the data, and the application of visualization techniques to create informative and visually appealing representations. Data preprocessing often includes steps such as normalization, interpolation, and the handling of missing values, which are crucial for ensuring the accuracy and reliability of the visualizations.

## Gaps in Current Research

Despite the significant advancements in environmental data visualization, several gaps remain. One major limitation is the lack of interactivity in many existing visualizations, which can hinder user engagement and exploration of the data. Static maps and graphs, while informative, do not allow users to interact with the data, such as selecting different parameters or viewing changes over time. This limits the ability to uncover deeper insights and trends.

Additionally, there is often a limited use of 3D modelling in environmental data visualization. While 2D maps provide valuable information, they can fall short in representing the complexity of environmental data, especially in regions with varied topography like Crete. 3D visualizations can offer a more immersive experience, allowing users to see how environmental parameters vary across different elevations and landscapes.

Another gap is the integration of multiple visualization techniques into a single, cohesive platform. Many projects focus on one type of visualization, such as static maps or time-series plots, but do not combine these approaches to provide a comprehensive view of the data. This fragmented approach can limit the overall understanding of environmental phenomena.

My thesis aims to address these gaps by integrating multiple visualization techniques, including static plots, interactive maps, 3D models, and animations. By leveraging Python libraries such as Folium, Matplotlib, and Plotly, I will create a dynamic and interactive platform for visualizing environmental data for Crete. This approach will enhance user engagement, provide a more comprehensive understanding of the data, and enable the exploration of environmental trends over time and across different spatial dimensions.

# Methodology

The methodology section outlines the overall approach taken to collect, process, and visualize the environmental data for Crete. This includes details about data sources, processing techniques, and the tools used for creating the visualizations.

## Data Collection

The environmental data was sourced from the ECMWF ERA5-Land reanalysis dataset. This dataset provides comprehensive data on various environmental parameters such as temperature, precipitation, soil moisture, surface pressure, and wind speed. The data covers the period from 1990 to 2020, ensuring a robust longitudinal study. The dataset is available in GeoTIFF format, which is well-suited for spatial analysis and visualization.

### Data Processing

Data processing involved several key steps to ensure the data was ready for visualization:

1. **Reprojection**: All data was reprojected to a common coordinate reference system (EPSG:4326) to ensure consistency across different datasets.
2. **Normalization**: Data values were normalized to a common scale to facilitate comparison across different parameters and years.
3. **Handling Missing Values**: Special attention was given to handling missing values. No-data values in the datasets were identified and appropriately managed to prevent any distortions in the visualizations.
4. **Conversion to PNG**: To streamline the visualization process, the GeoTIFF files were converted to PNG format.

### Visualization Tools and Techniques

Various tools and techniques were employed to create different types of visualizations:

1. **Matplotlib**: Used for creating static plots due to its wide range of plotting functionalities and customization options.
2. **Folium**: Utilized for generating interactive maps, leveraging the power of Leaflet.js for creating web-based maps.
3. **PyVista**: Employed for creating 3D models, working with VTK to handle mesh and point cloud data.
4. **ImageIO**: Used for creating animated GIFs, allowing for the visualization of changes over time.

Each tool was selected based on its strengths and suitability for specific visualization tasks.

### 3D Model Creation and Data Projection

3D models were created using PyVista, which allowed for the visualization of environmental data over a digital elevation model (DEM) of Crete. The DEM data was processed to create a mesh grid, and environmental data was overlaid as scalar fields. This approach provided an intuitive understanding of the spatial distribution of environmental parameters.

## Challenges and Solutions

Several challenges were encountered during the project:

1. **Data Quality**: Ensuring data accuracy and handling missing values were critical. This was managed through careful data cleaning and normalization.
2. **Performance**: Efficiently handling large datasets required optimization techniques to maintain smooth performance, particularly for interactive maps.
3. **Integration**: Combining different visualization methods into a cohesive platform involved careful planning to ensure compatibility and a seamless user experience.
4. **User Interaction**: Designing intuitive interfaces for interactive maps was crucial for easy navigation and comprehension.

## Summary of Key Resources and Methods

The key resources and methods utilized in this thesis included:

* **Data Sources**: ECMWF ERA5-Land reanalysis dataset, Copernicus DEM GLO-30.
* **Programming Languages and Libraries**: Python, Rasterio, Numpy, Matplotlib, Folium, PyVista, ImageIO, HTML, CSS.
* **Techniques**: Data normalization, re-projection, 3D modeling, interactive map creation, and GIF animation.

# Implementation

The implementation chapter details the practical steps and methodologies used to visualize the environmental data for Crete. This includes data collection, processing, creation of various visualizations, and integration into an interactive map.

## Visualization Process

The visualization process was meticulously structured into several stages, each essential for transforming raw environmental data into meaningful visual insights.

### Data Collection and Preparation

The data collection phase involved acquiring environmental data from the ECMWF ERA5-Land reanalysis dataset. This dataset, spanning from 1990 to 2020, includes daily data on temperature, precipitation, soil moisture, surface pressure, and wind speed, stored in GeoTIFF format.

Script to download the 5 bands for Crete from 1990 to 2020 and save them in Google Drive:

// Define the region of interest (Crete)  
var aoi = ee.Geometry.Polygon(  
 [[[23.34879154216733, 35.69108822688866],  
 [23.34879154216733, 34.88851141701856],  
 [26.71610111247983, 34.88851141701856],  
 [26.71610111247983, 35.69108822688866]]], null, false);  
   
  
var dataset = ee.ImageCollection('ECMWF/ERA5\_LAND/DAILY\_AGGR');  
// Define a function to export data for each year and band  
var exportData = function(year, band, description, folder) {  
 var startDate = ee.Date.fromYMD(year, 1, 1);  
 var endDate = startDate.advance(1, 'year');  
 var yearlyData = dataset.filterDate(startDate, endDate).select(band).mean().clip(aoi);  
   
 Export.image.toDrive({  
 image: yearlyData,  
 description: description + '\_' + year,  
 folder: folder,  
 scale: 1000,  
 region: aoi,  
 fileFormat: 'GeoTIFF'  
 });  
};  
  
// Years and bands to export  
var years = ee.List.sequence(2000,2020);  
var bands = [  
 {band: 'temperature\_2m', description: 'Crete\_Temperature\_2m', folder: 'Crete\_Temperature'},  
 {band: 'total\_precipitation\_sum', description: 'Crete\_Total\_Precipitation', folder: 'Crete\_Precipitation'},  
 {band: 'volumetric\_soil\_water\_layer\_1', description: 'Crete\_Soil\_Moisture', folder: 'Crete\_Soil\_Moisture'},  
 {band: 'u\_component\_of\_wind\_10m', description: 'Crete\_Wind\_U', folder: 'Crete\_Wind\_U'},  
 {band: 'surface\_pressure', description: 'Crete\_Surface\_Pressure', folder: 'Crete\_Surface\_Pressure'}  
];  
  
// Loop over each year and band to export data  
years.getInfo().forEach(function(year) {  
 bands.forEach(function(band) {  
 exportData(year, band.band, band.description, band.folder);  
 });  
});

Script to download the Crete DEM file from Copernicus dataset and store it in Google Drive:

// Define the region of interest (AOI)  
var aoi = ee.Geometry.Polygon(  
 [[[23.34879154216733, 35.69108822688866],  
 [23.34879154216733, 34.88851141701856],  
 [26.71610111247983, 34.88851141701856],  
 [26.71610111247983, 35.69108822688866]]], null, false);  
  
// Load the DEM dataset and clip it to the AOI  
var dataset = ee.ImageCollection('COPERNICUS/DEM/GLO30');  
var elevation = dataset.select('DEM').mosaic().clip(aoi);  
  
// Visualization parameters  
var elevationVis = {  
 min: 0.0,  
 max: 1000.0,  
 palette: ['0000ff','00ffff','ffff00','ff0000','ffffff'],  
};  
  
// Export the DEM to Google Drive  
Export.image.toDrive({  
 image: elevation,  
 description: 'Crete\_DEM',  
 scale: 30,  
 region: aoi,  
 fileFormat: 'GeoTIFF'  
});  
  
// Add the DEM layer to the map  
Map.setCenter(25.0, 35.2, 8); // Center the map on Crete  
Map.addLayer(elevation, elevationVis, 'DEM');

### Data Processing

The data processing phase was critical to ensure the data's accuracy and consistency for visualization. The steps included:

1. Reprojection: Converting all datasets to a common coordinate reference system (EPSG:4326) to ensure spatial consistency.

2. Normalization: Scaling the data values to a uniform range to facilitate comparison across different parameters and years.

3. Handling Missing Values: Identifying and managing no-data values to prevent distortions in the visualizations.

4. Conversion to PNG: Transforming the GeoTIFF files into PNG format to streamline the visualization process.

Normalisation of dataset:

import os  
import pandas as pd  
import numpy as np  
import rasterio  
from rasterio.warp import reproject, Resampling  
from rasterio.plot import show  
import matplotlib.pyplot as plt  
  
# Define file paths and CRS details  
base\_path = 'c:/Users/nboub/Desktop/'  
dem\_path = os.path.join(base\_path, 'crete\_dem.tif')  
  
# Load the DEM file  
print(f"Loading DEM file from {dem\_path}")  
dem\_data = rasterio.open(dem\_path)  
  
# Define the destination CRS and transform based on the DEM  
dest\_crs = dem\_data.crs  
dest\_transform = dem\_data.transform  
print(f"Destination CRS: {dest\_crs}")  
  
# Function to read and process each year's data  
def process\_yearly\_data(year):  
    try:  
        # Define file paths for the downloaded data  
        temperature\_path = os.path.join(base\_path, f'Crete\_Temperature/Crete\_Temperature\_2m\_{year}.tif')  
        total\_Precipitation\_path = os.path.join(base\_path, f'Crete\_total\_Precipitation/Crete\_Total\_total\_Precipitation\_{year}.tif')  
        soil\_moisture\_path = os.path.join(base\_path, f'Crete\_Soil\_Moisture/Crete\_Soil\_Moisture\_{year}.tif')  
        surface\_pressure\_path = os.path.join(base\_path, f'Crete\_Surface\_Pressure/Crete\_Surface\_Pressure\_{year}.tif')  
        wind\_u\_path = os.path.join(base\_path, f'Crete\_Wind\_U/Crete\_Wind\_U\_{year}.tif')  
  
        # Check if files exist  
        file\_paths = [temperature\_path, total\_Precipitation\_path, soil\_moisture\_path, surface\_pressure\_path, wind\_u\_path]  
        for path in file\_paths:  
            if not os.path.exists(path):  
                print(f"File not found: {path}")  
                return {'year': year, 'temperature': np.nan, 'total\_Precipitation': np.nan, 'soil\_moisture': np.nan, 'surface\_pressure': np.nan, 'wind\_u': np.nan}  
  
        # Load the temperature data  
        print(f"Loading temperature data from {temperature\_path}")  
        temperature\_data = rasterio.open(temperature\_path)  
        temperature\_array = temperature\_data.read(1)  
  
        # Reproject the temperature data to match the DEM  
        print("Reprojecting temperature data...")  
        reprojected\_temperature = np.empty\_like(dem\_data.read(1))  
        reproject(  
            source=temperature\_array,  
            destination=reprojected\_temperature,  
            src\_transform=temperature\_data.transform,  
            src\_crs=temperature\_data.crs,  
            dst\_transform=dest\_transform,  
            dst\_crs=dest\_crs,  
            resampling=Resampling.nearest  
        )  
  
        # Calculate mean values  
        temperature\_mean = np.mean(reprojected\_temperature)  
        total\_Precipitation\_mean = np.mean(rasterio.open(total\_Precipitation\_path).read(1))  
        soil\_moisture\_mean = np.mean(rasterio.open(soil\_moisture\_path).read(1))  
        surface\_pressure\_mean = np.mean(rasterio.open(surface\_pressure\_path).read(1))  
        wind\_u\_mean = np.mean(rasterio.open(wind\_u\_path).read(1))  
  
        print(f"Year {year}: Temperature Mean = {temperature\_mean}, total\_Precipitation Mean = {total\_Precipitation\_mean}, Soil Moisture Mean = {soil\_moisture\_mean}, Surface Pressure Mean = {surface\_pressure\_mean}, Wind U Mean = {wind\_u\_mean}")  
  
        # Return the processed data  
        return {  
            'year': year,  
            'temperature': temperature\_mean,  
            'total\_Precipitation': total\_Precipitation\_mean,  
            'soil\_moisture': soil\_moisture\_mean,  
            'surface\_pressure': surface\_pressure\_mean,  
            'wind\_u': wind\_u\_mean  
        }  
    except Exception as e:  
        print(f"Error processing data for year {year}: {e}")  
        return {  
            'year': year,  
            'temperature': np.nan,  
            'total\_Precipitation': np.nan,  
            'soil\_moisture': np.nan,  
            'surface\_pressure': np.nan,  
            'wind\_u': np.nan  
        }  
  
# Process data for all years and store results  
all\_data = []  
for year in range(2000, 2021):  
    data = process\_yearly\_data(year)  
    all\_data.append(data)  
  
# Convert results to a DataFrame  
df = pd.DataFrame(all\_data)  
  
# Normalize the data  
def normalize\_data(df):  
    df\_normalized = df.copy()  
    for column in df.columns:  
        if column != 'year':  
            df\_normalized[column] = (df[column] - df[column].min()) / (df[column].max() - df[column].min())  
    return df\_normalized  
  
df\_normalized = normalize\_data(df)  
  
print(df\_normalized)  
  
# Save the normalized data  
df\_normalized.to\_csv('normalized\_crete\_climate\_data.csv', index=False)

### Visualization Creation

The creation of visualizations involved several tools and methods, each chosen for its strengths in handling specific visualization tasks.

1. Static Plots: Utilized Matplotlib to create static visualizations for various environmental parameters across different years. These plots provide a clear and immediate understanding of the data's spatial distribution.

import os  
import numpy as np  
import pandas as pd  
import rasterio  
import matplotlib.pyplot as plt  
from matplotlib.colors import Normalize  
from rasterio.plot import show  
  
# Define paths  
base\_paths = {  
    'temperature\_2m': 'C:/Users/nboub/Pictures/Crete\_Temperature',  
    'total\_Precipitation': 'C:/Users/nboub/Pictures/Crete\_total\_Precipitation',  
    'soil\_Moisture': 'C:/Users/nboub/Pictures/Crete\_Soil\_Moisture',  
    'surface\_Pressure': 'C:/Users/nboub/Pictures/Crete\_Surface\_Pressure',  
    'wind\_U': 'C:/Users/nboub/Pictures/Crete\_Wind\_U'  
}  
dem\_path = 'C:/Users/nboub/Desktop/crete\_dem.tif'  
  
# Define units for each band  
units = {  
    'temperature\_2m': 'K',  
    'total\_Precipitation': 'mm',  
    'soil\_Moisture': 'm³/m³',  
    'surface\_Pressure': 'Pa',  
    'wind\_U': 'm/s'  
}  
  
# Load the DEM file  
print(f"Loading DEM file from {dem\_path}")  
with rasterio.open(dem\_path) as dem\_data:  
    dem\_array = dem\_data.read(1)  
    dest\_crs = dem\_data.crs  
    dest\_transform = dem\_data.transform  
print(f"Destination CRS: {dest\_crs}")  
  
# Function to read and verify each year's data for a given band  
def process\_yearly\_data(year, band, base\_path):  
    try:  
        # Define file path for the data  
        data\_path = os.path.join(base\_path, f'Crete\_{band.capitalize()}\_{year}.tif')  
  
        # Check if file exists  
        if not os.path.exists(data\_path):  
            print(f"File not found: {data\_path}")  
            return {'year': year, band: np.nan, f'{band}\_array': np.nan}  
  
        # Load the data  
        print(f"Loading {band} data from {data\_path}")  
        with rasterio.open(data\_path) as data:  
            data\_array = data.read(1)  
            metadata = data.meta  
  
            # Debugging: Check the metadata and some data values  
            print(f"Metadata for {year} {band}: {metadata}")  
            print(f"{band.capitalize()} data sample for {year}: {data\_array[0:5, 0:5]}")  
  
            # Debugging: Check the data  
            min\_val = np.nanmin(data\_array)  
            max\_val = np.nanmax(data\_array)  
            mean\_val = np.nanmean(data\_array)  
            print(f"{band.capitalize()} data for {year}: min={min\_val}, max={max\_val}, mean={mean\_val}")  
  
            # Return the processed data  
            return {  
                'year': year,  
                band: mean\_val,  
                f'{band}\_array': data\_array  # Store the array for later plotting  
            }  
    except Exception as e:  
        print(f"Error processing data for year {year} band {band}: {e}")  
        return {  
            'year': year,  
            band: np.nan,  
            f'{band}\_array': np.nan  
        }  
  
# Bands to process  
bands = ['temperature\_2m', 'total\_Precipitation', 'soil\_Moisture', 'surface\_Pressure', 'wind\_U']  
  
# Process data for all years and all bands  
all\_data = {band: [] for band in bands}  
global\_min\_max = {band: {'min': np.inf, 'max': -np.inf} for band in bands}  
  
for year in range(1990, 2021):  
    for band in bands:  
        data = process\_yearly\_data(year, band, base\_paths[band])  
        all\_data[band].append(data)  
          
        # Update global min and max for the band  
        if not np.isnan(data[f'{band}\_array']).all():  
            global\_min\_max[band]['min'] = min(global\_min\_max[band]['min'], np.nanmin(data[f'{band}\_array']))  
            global\_min\_max[band]['max'] = max(global\_min\_max[band]['max'], np.nanmax(data[f'{band}\_array']))  
  
# Convert results to DataFrames  
df\_all = {band: pd.DataFrame(all\_data[band]) for band in bands}  
  
# Normalize the data  
def normalize\_data(df):  
    df\_normalized = df.copy()  
    for column in df.columns:  
        if column != 'year' and not column.endswith('\_array'):  
            df\_normalized[column] = (df[column] - df[column].min()) / (df[column].max() - df[column].min())  
    return df\_normalized  
  
df\_all\_normalized = {band: normalize\_data(df\_all[band]) for band in bands}  
  
# Save the normalized data  
for band, df in df\_all\_normalized.items():  
    output\_csv\_path = os.path.join('C:/Users/nboub/Desktop', f'normalized\_crete\_{band}\_data.csv')  
    df.to\_csv(output\_csv\_path, index=False)  
    print(f"Normalized {band} data saved to {output\_csv\_path}")  
  
# Plotting the data and saving to files  
def plot\_and\_save\_data(df, band, cmap, global\_min, global\_max, unit):  
    plot\_dir = os.path.join('C:/Users/nboub/Desktop/Plots', band)  
    os.makedirs(plot\_dir, exist\_ok=True)  
    for index, row in df.iterrows():  
        data\_array = row[f'{band}\_array']  # Access the stored data array  
        if np.isnan(data\_array).all():  
            print(f"Skipping plot for {band} in {row['year']} due to all NaN values")  
            continue  
        plt.figure(figsize=(10, 6))  
        plt.imshow(data\_array, cmap=cmap, norm=Normalize(vmin=global\_min, vmax=global\_max))  
        cbar = plt.colorbar()  
        cbar.set\_label(f'Unit: {unit}')  
        plt.title(f'{band.capitalize()} Data for {row["year"]}')  
        plot\_path = os.path.join(plot\_dir, f'{band}\_data\_{row["year"]}.png')  
        plt.savefig(plot\_path)  
        plt.close()  
        print(f'Saved plot to {plot\_path}')  
  
# Define color maps for each band  
cmap\_dict = {  
    'temperature\_2m': 'hot',  
    'total\_Precipitation': 'Blues',  
    'soil\_Moisture': 'Greens',  
    'surface\_Pressure': 'Oranges',  
    'wind\_U': 'Purples'  
}  
  
# Plot the data for each band and save to files  
for band in bands:  
    plot\_and\_save\_data(df\_all[band], band, cmap\_dict[band], global\_min\_max[band]['min'], global\_min\_max[band]['max'], units[band])

2. Simple Line Graphs: Created line graphs using Matplotlib to visualize temporal trends in numerical data. These graphs are crucial for identifying long-term changes and trends, such as those related to climate change.

Example of Temperature graph code:

from matplotlib import pyplot as plt  
import seaborn as sns  
def \_plot\_series(series, series\_name, series\_index=0):  
  palette = list(sns.palettes.mpl\_palette('Dark2'))  
  xs = series['year']  
  ys = series['temperature\_2m']  
  
  plt.plot(xs, ys, label=series\_name, color=palette[series\_index % len(palette)])  
  
fig, ax = plt.subplots(figsize=(10, 5.2), layout='constrained')  
df\_sorted = df.sort\_values('year', ascending=True)  
\_plot\_series(df\_sorted, '')  
sns.despine(fig=fig, ax=ax)  
plt.xlabel('year')  
\_ = plt.ylabel('temperature\_2m')

3. 3D Models: PyVista was employed to develop 3D models that overlay environmental data on a digital elevation model (DEM) of Crete. This method offers a comprehensive view of the spatial variations in environmental parameters.

import os  
import numpy as np  
import rioxarray as riox  
import pyvista as pv  
import time  
  
# Paths to your data files  
base\_folder = 'C:/Users/nboub/Pictures'  
folders = {  
    'Crete\_Temperature': 'Crete\_Temperature',  
    'Crete\_Precipitation': 'Crete\_total\_Precipitation',  
    'Crete\_Soil\_Moisture': 'Crete\_Soil\_Moisture',  
    'Crete\_Wind\_U': 'Crete\_Wind\_U',  
    'Crete\_Surface\_Pressure': 'Crete\_Surface\_Pressure'  
}  
  
# Color maps for each data type  
color\_maps = {  
    'Crete\_Temperature': 'coolwarm',  
    'Crete\_Precipitation': 'Blues',  
    'Crete\_Soil\_Moisture': 'Greens',  
    'Crete\_Wind\_U': 'Purples',  
    'Crete\_Surface\_Pressure': 'Oranges'  
}  
  
# Define units for each band  
units = {  
    'Crete\_Temperature': 'K',  
    'Crete\_Precipitation': 'mm',  
    'Crete\_Soil\_Moisture': 'm³/m³',  
    'Crete\_Wind\_U': 'm/s',  
    'Crete\_Surface\_Pressure': 'Pa'  
}  
  
# Load the DEM data  
print("Loading DEM data...")  
dem\_path = 'C:/Users/nboub/Desktop/crete\_dem.tif'  
dem\_data = riox.open\_rasterio(dem\_path)  
dem\_data = dem\_data[0]  # Select the first band  
print(f"DEM data shape: {dem\_data.shape}")  
  
# Function to load and resample data  
def load\_and\_resample(path, dem\_data):  
    print(f"Loading data from {path}...")  
    data = riox.open\_rasterio(path)  
    data = data[0]  # Select the first band  
    print(f"Original data shape: {data.shape}")  
    data = data.rio.reproject\_match(dem\_data)  
    print(f"Resampled data shape: {data.shape}")  
    return np.asarray(data)  
  
# Create a mesh grid for the DEM  
print("Creating mesh grid...")  
x, y = np.meshgrid(dem\_data['x'], dem\_data['y'])  
print(f"Mesh grid shapes - x: {x.shape}, y: {y.shape}")  
  
# Set the z values and create a StructuredGrid  
print("Creating StructuredGrid...")  
z = np.zeros\_like(x)  
mesh = pv.StructuredGrid(x.astype(np.float32), y.astype(np.float32), z.astype(np.float32))  
print(f"StructuredGrid created with {mesh.n\_points} points.")  
  
# Assign Elevation Values  
print("Assigning elevation values...")  
mesh["Elevation"] = np.asarray(dem\_data).ravel(order='F')  
print("Elevation values assigned.")  
  
# Warp the mesh by scalar to visualize the terrain  
print("Warping the mesh by scalar...")  
topo = mesh.warp\_by\_scalar(scalars="Elevation", factor=0.00005)  # Adjust the factor as needed  
print("Mesh warped by scalar.")  
  
# Function to get global min and max values for each data type  
def get\_global\_min\_max(folder\_full\_path):  
    global\_min = np.inf  
    global\_max = -np.inf  
    for file\_name in os.listdir(folder\_full\_path):  
        if file\_name.endswith('.tif'):  
            file\_path = os.path.join(folder\_full\_path, file\_name)  
            data = load\_and\_resample(file\_path, dem\_data)  
            global\_min = min(global\_min, np.nanmin(data))  
            global\_max = max(global\_max, np.nanmax(data))  
    return global\_min, global\_max  
  
# Function to plot data and save the output  
def plot\_data(topo, data, title, cmap, output\_folder, global\_min, global\_max, unit):  
    print(f"Overlaying {title} data...")  
    raveled\_data = data.ravel(order='F')  
    topo[title] = raveled\_data  
    print(f"{title} data overlayed.")  
     
    print(f"Plotting the 3D terrain with {title} overlay...")  
    p = pv.Plotter(off\_screen=True)  
    p.add\_mesh(topo, scalars=title, cmap=cmap, scalar\_bar\_args={'title': f'{title} ({unit})', 'label\_font\_size': 10})  
    p.set\_background(color='white')  
    p.show\_bounds(grid='back', location='outer', ticks='both', font\_size=7)  # Move the grid to the back  
  
    # Adjust the camera position  
    p.camera\_position = 'xy'  
    p.camera.azimuth = 320  # Rotate around the vertical axis  
    p.camera.elevation = 20  # Rotate around the horizontal axis to view from above  
    p.camera.roll = 0 # Adjust roll to ensure north is up  
  
    # Ensure the plot is fully rendered before taking the screenshot  
    p.show(auto\_close=False)  
    time.sleep(1)  # Add a short delay to ensure the render window is updated  
    output\_path = os.path.join(output\_folder, f"{title}.png")  
    p.screenshot(output\_path)  
    p.close()  
    print(f"Plot saved for {title} at {output\_path}.")  
  
# Process each folder and file  
for folder\_name, folder\_path in folders.items():  
    folder\_full\_path = os.path.join(base\_folder, folder\_path)  
    output\_folder = os.path.join(base\_folder, f"{folder\_path}\_Output")  
    os.makedirs(output\_folder, exist\_ok=True)  
    cmap = color\_maps[folder\_name]  
    unit = units[folder\_name]  
     
    # Get global min and max values for consistent color bar  
    global\_min, global\_max = get\_global\_min\_max(folder\_full\_path)  
     
    for file\_name in os.listdir(folder\_full\_path):  
        if file\_name.endswith('.tif'):  
            file\_path = os.path.join(folder\_full\_path, file\_name)  
            data = load\_and\_resample(file\_path, dem\_data)  
            title = os.path.splitext(file\_name)[0]  
            plot\_data(topo, data, title, cmap, output\_folder, global\_min, global\_max, unit)

4. GIF Animations: To visualize changes over time, ImageIO was used to create animated GIFs from the series of static plots. This dynamic representation helps in understanding temporal trends in the data.

import os  
from PIL import Image  
  
# Define paths  
base\_plot\_path = 'C:/Users/nboub/Desktop/Plots'  
output\_gif\_path = 'C:/Users/nboub/Desktop/GIFs'  
  
# Ensure the output directory exists  
os.makedirs(output\_gif\_path, exist\_ok=True)  
  
# Define the bands  
bands = ['temperature\_2m', 'total\_Precipitation', 'soil\_Moisture', 'surface\_Pressure', 'wind\_U']  
  
# Define the speed of the GIF (duration in milliseconds between frames)  
frame\_duration = 400 # 1 second per frame  
  
# Function to create a GIF for a given band  
def create\_gif(band):  
    plot\_dir = os.path.join(base\_plot\_path, band)  
    output\_gif\_file = os.path.join(output\_gif\_path, f'{band}.gif')  
  
    # Get all PNG files in the directory and sort them  
    image\_files = sorted([os.path.join(plot\_dir, file) for file in os.listdir(plot\_dir) if file.endswith('.png')])  
  
    # Load images  
    images = [Image.open(image\_file) for image\_file in image\_files]  
  
    # Save as GIF  
    images[0].save(output\_gif\_file, save\_all=True, append\_images=images[1:], duration=frame\_duration, loop=0)  
    print(f'Saved GIF for {band} to {output\_gif\_file}')  
  
# Create GIFs for each band  
for band in bands:  
    create\_gif(band)

5. Interactive Maps: Folium was chosen to create interactive maps. These maps allow users to zoom in on specific regions and view data overlays, providing an engaging and detailed exploration of the data.

import folium  
  
# Coordinates for Greece and Crete  
greece\_coords = [39.0742, 21.8243]  
crete\_coords = [35.2401, 24.8093]  
  
# Create a map centered on Greece  
m = folium.Map(location=greece\_coords, zoom\_start=6)  
  
# Add a marker on Crete with a popup that includes a zoom button, dropdown menus, and a slider  
popup\_html = """  
<style>  
    .zoom-button {  
        background-color: #b6c1ff;   
        border: none;  
        color: white;  
        padding: 10px 24px;  
        text-align: center;  
        text-decoration: none;  
        display: inline-block;  
        font-size: 16px;  
        margin: 4px 2px;  
        transition-duration: 0.4s;  
        cursor: pointer;  
        border-radius: 8px;  
    }  
  
    .zoom-button:hover {  
        background-color: white;  
        color: black;  
    }  
  
    .popup-content {  
        font-family: 'Arial', sans-serif;  
        font-size: 14px;  
        color: #333;  
        text-align: left;  
        padding: 10px;  
    }  
  
    .popup-content label {  
        font-weight: bold;  
        margin-bottom: 5px;  
        display: block;  
    }  
  
    .popup-content select, .popup-content input {  
        width: 100%;  
        margin-bottom: 10px;  
        padding: 5px;  
        border-radius: 4px;  
        border: 1px solid #ccc;  
    }  
  
    .popup-content input[type="range"] {  
        -webkit-appearance: none;  
        width: 100%;  
        height: 8px;  
        border-radius: 5px;  
        background: #d3d3d3;  
        outline: none;  
        opacity: 0.7;  
        -webkit-transition: .2s;  
        transition: opacity .2s;  
    }  
  
    .popup-content input[type="range"]::-webkit-slider-thumb {  
        -webkit-appearance: none;  
        appearance: none;  
        width: 25px;  
        height: 25px;  
        border-radius: 50%;  
        background: #4CAF50;  
        cursor: pointer;  
    }  
  
    .popup-content input[type="range"]::-moz-range-thumb {  
        width: 25px;  
        height: 25px;  
        border-radius: 50%;  
        background: #4CAF50;  
        cursor: pointer;  
    }  
  
    .popup-content output {  
        display: block;  
        margin-top: 5px;  
        font-weight: bold;  
    }  
</style>  
<div class="popup-content">  
    <button class="zoom-button" onclick="zoomToCrete()">Zoom to Crete</button>  
    <br>  
    <label for="bandSelect">Select Band:</label>  
    <select id="bandSelect">  
        <option value="Temperature\_2m">Temperature</option>  
        <option value="Total\_Precipitation">Precipitation</option>  
        <option value="Soil\_Moisture">Soil Moisture</option>  
        <option value="Surface\_Pressure">Surface Pressure</option>  
        <option value="Wind\_U">Wind U</option>  
    </select>  
    <br>  
    <label for="yearSlider">Select Year:</label>  
    <input type="range" id="yearSlider" name="yearSlider" min="1990" max="2020" value="2000" oninput="this.nextElementSibling.value = this.value">  
    <output>2000</output>  
</div>  
"""  
popup = folium.Popup(popup\_html, max\_width=300)  
  
marker = folium.Marker(  
    location=crete\_coords,  
    popup=popup,  
    icon=folium.Icon(color="lightred", icon="info-sign")  
).add\_to(m)  
  
# Additional markers for other locations  
athens\_coords = [37.9838, 23.7275]  
thessaloniki\_coords = [40.6401, 22.9444]  
  
folium.Marker(  
    location=athens\_coords,  
    popup="Athens",  
    icon=folium.Icon(color="beige", icon="info-sign")  
).add\_to(m)  
  
folium.Marker(  
    location=thessaloniki\_coords,  
    popup="Thessaloniki",  
    icon=folium.Icon(color="lightblue", icon="info-sign")  
).add\_to(m)  
  
# Get the map ID  
map\_id = m.get\_name()  
  
# Add custom JavaScript to handle the zoom and interactions  
zoom\_js = f"""  
<script>  
    var overlayLayer;  
    var colorbarLayer;  
  
    document.addEventListener('DOMContentLoaded', function() {{  
        window.map = {map\_id};  // Ensure the map is available in the global scope  
    }});  
  
    function zoomToCrete() {{  
        map.setView([35.2401, 24.8093], 8);  // Adjusted zoom level  
        document.getElementById('bandSelect').addEventListener('change', updateMap);  
        document.getElementById('yearSlider').addEventListener('input', updateMap);  
    }}  
  
    function updateMap() {{  
        var band = document.getElementById('bandSelect').value;  
        var year = document.getElementById('yearSlider').value;  
        console.log('Selected Band:', band, 'Selected Year:', year);  
  
        // Example: Update the overlay layer (Assuming you have a server to serve .png files)  
        var overlayUrl = `http://localhost:8000/Crete\_${{band}}\_${{year}}.png`;  
        console.log('Overlay URL:', overlayUrl);  
          
        // Remove the existing overlay layer if it exists  
        if (overlayLayer) {{  
            map.removeLayer(overlayLayer);  
        }}  
          
        // Add the new overlay layer  
        var bounds = [[34.8, 23.3], [35.8, 26.7]];  
        overlayLayer = L.imageOverlay(overlayUrl, bounds, {{ opacity: 0.6 }});  
        overlayLayer.addTo(map);  
  
        // Update the color bar  
        if (colorbarLayer) {{  
            map.removeLayer(colorbarLayer);  
        }}  
  
        var colorbarUrl;  
        switch(band) {{  
            case 'Temperature\_2m':  
                colorbarUrl = 'http://localhost:8000/colorbar\_Temperature\_2m.png';  
                break;  
            case 'Total\_Precipitation':  
                colorbarUrl = 'http://localhost:8000/colorbar\_Total\_Precipitation.png';  
                break;  
            case 'Soil\_Moisture':  
                colorbarUrl = 'http://localhost:8000/colorbar\_Soil\_Moisture.png';  
                break;  
            case 'Surface\_Pressure':  
                colorbarUrl = 'http://localhost:8000/colorbar\_Surface\_Pressure.png';  
                break;  
            case 'Wind\_U':  
                colorbarUrl = 'http://localhost:8000/colorbar\_Wind\_U.png';  
                break;  
        }}  
  
       var colorbarBounds = [[34.8, 22.9], [36.0, 23.3]];  // Adjusted bounds for a bigger color bar  
        colorbarLayer = L.imageOverlay(colorbarUrl, colorbarBounds, {{ opacity: 1.0 }});  
        colorbarLayer.addTo(map);  
    }}  
</script>  
"""  
  
# Add the JavaScript to the map  
m.get\_root().html.add\_child(folium.Element(zoom\_js))  
  
# Save the map  
m.save("C:/Users/nboub/Pictures/Data1/greece\_map.html")  
  
print("Map saved as greece\_map.html. Open this file in a web browser to view it.")

### Integration and Presentation

The final stage involved integrating all visualizations into a cohesive platform. This included:

- Embedding the visualizations into a web-based interface for easy access and interaction.

- Implementing a server to serve the PNG files and interactive map functionalities.

- Ensuring that the visualizations are responsive and user-friendly, enhancing the overall user experience.

TIF to PNG conversion:

import rasterio  
import numpy as np  
from PIL import Image  
import matplotlib.pyplot as plt  
from matplotlib.colors import Normalize  
  
# Color maps for each band  
color\_maps = {  
 'Temperature\_2m': 'hot',  
 'Total\_Precipitation': 'Blues',  
 'Soil\_Moisture': 'Greens',  
 'Surface\_Pressure': 'Oranges',  
 'Wind\_U': 'Purples'  
}  
  
def apply\_color\_map\_with\_transparency(array, cmap):  
 norm = Normalize(vmin=np.nanmin(array), vmax=np.nanmax(array))  
 colormap = plt.get\_cmap(cmap)  
 rgba\_img = colormap(norm(array))  
 rgba\_img[:, :, 3] = ~np.isnan(array) # Set alpha channel: 1 for data, 0 for NaN  
 rgba\_img = (rgba\_img \* 255).astype(np.uint8) # Convert to 8-bit per channel  
 return rgba\_img  
  
def convert\_tif\_to\_png(tif\_path, png\_path, cmap):  
 with rasterio.open(tif\_path) as src:  
 array = src.read(1)  
 array[array == src.nodata] = np.nan  
  
 # Apply color map with transparency  
 color\_mapped\_array = apply\_color\_map\_with\_transparency(array, cmap)  
  
 # Save the color-mapped array as a PNG file with transparency  
 img = Image.fromarray(color\_mapped\_array, 'RGBA')  
 img.save(png\_path)  
  
# Convert all your TIF files to PNG with color mapping and transparency  
for band, cmap in color\_maps.items():  
 for year in range(1990, 2021):  
 tif\_path = f"C:/Users/nboub/Pictures/Data/Crete\_{band}\_{year}.tif"  
 png\_path = f"C:/Users/nboub/Pictures/Data1/Crete\_{band}\_{year}.png"  
 convert\_tif\_to\_png(tif\_path, png\_path, cmap)

Server initialisation:

import http.server  
import socketserver  
  
PORT = 8000  
DIRECTORY = "C:/Users/nboub/Pictures/Data1"  
  
class MyHttpRequestHandler(http.server.SimpleHTTPRequestHandler):  
    def \_\_init\_\_(self, \*args, \*\*kwargs):  
        super().\_\_init\_\_(\*args, directory=DIRECTORY, \*\*kwargs)  
  
handler\_object = MyHttpRequestHandler  
  
with socketserver.TCPServer(("", PORT), handler\_object) as httpd:  
    print(f"Serving at port {PORT}")  
    httpd.serve\_forever()

Color bar creation:

import matplotlib.pyplot as plt  
import numpy as np  
  
# Define the color maps and their ranges for each band  
color\_maps = {  
    'Temperature\_2m': ('hot', 270, 310, 'K'),  
    'Total\_Precipitation': ('Blues', 0, 300, 'mm'),  
    'Soil\_Moisture': ('Greens', 0, 1, 'm³/m³'),  
    'Surface\_Pressure': ('Oranges', 950, 1050, 'hPa'),  
    'Wind\_U': ('Purples', 0, 15, 'm/s')  
}  
  
def create\_colorbar(cmap\_name, vmin, vmax, label, filename):  
    fig, ax = plt.subplots(figsize=(2, 8), dpi=100, subplot\_kw={'frame\_on': False})  # Increased height to make it longer  
    norm = plt.Normalize(vmin=vmin, vmax=vmax)  
    fig.patch.set\_alpha(0.0)  # Set figure background to transparent  
    fig.subplots\_adjust(left=0.3, right=0.8, top=0.9, bottom=0.1)  
  
    cb = plt.colorbar(plt.cm.ScalarMappable(norm=norm, cmap=cmap\_name), cax=ax, orientation='vertical')  
    cb.set\_label(label, fontsize=12)  
    cb.outline.set\_visible(False)  
    cb.ax.yaxis.set\_tick\_params(color='black')  
  
    plt.savefig(filename, transparent=True, bbox\_inches='tight', pad\_inches=0)  
    plt.close()  
  
for band, (cmap, vmin, vmax, unit) in color\_maps.items():  
    create\_colorbar(cmap, vmin, vmax, f'{band} ({unit})', f'colorbar\_{band}.png')

## Challenges

Several challenges were encountered during the implementation phase:

1. Data Quality: Ensuring the accuracy of the data and managing missing values were crucial. This was managed through rigorous data cleaning and normalization processes.

2. Performance: Efficiently handling large datasets required optimization techniques to maintain smooth performance, especially for interactive maps.

3. Integration: Combining various visualization methods into a cohesive and user-friendly platform required careful planning and execution.

4. User Interaction: Designing intuitive interfaces for the interactive maps was vital for user engagement and comprehension.

By systematically addressing these challenges, the project successfully created a robust and insightful visualization platform for environmental data in Crete.

# Results

This chapter presents the results derived from the visualized environmental data of Crete. The section is divided into key findings and examples of visualizations, providing both analytical insights and visual representations.

## Key Findings from Visualized Data

The visualizations produced a wealth of information regarding the environmental parameters of Crete over the past three decades. Key findings from the visualized data include:

1. **Temperature Trends**:
   * A clear upward trend in the average annual temperatures from 1990 to 2020 was observed.A graph showing the temperature of a number of data

     Description automatically generated with medium confidence
2. **Precipitation Patterns**:
   * The visualizations revealed a decreasing trend in annual precipitation, indicating potential drought conditions in recent years.
   * Spatial variability was prominent, with certain regions of Crete receiving significantly less rainfall over time.

|  |  |
| --- | --- |
|  |  |

1. **Soil Moisture Levels**:
   * Soil moisture levels showed significant fluctuations, correlating with precipitation patterns.
   * Regions with decreasing precipitation also exhibited declining soil moisture levels, impacting agriculture and vegetation.

|  |  |
| --- | --- |
| A graph of soil moisture over the years  Description automatically generated | A graph of soil moisture data  Description automatically generated |

1. **Surface Pressure Variations**:
   * The visualizations of surface pressure indicated relatively stable trends with minor fluctuations.

|  |  |
| --- | --- |
|  |  |

1. **Wind Speed Analysis**:
   * Wind speed data visualizations highlighted certain areas with consistently higher wind speeds, relevant for wind energy potential.

**A screenshot of a map

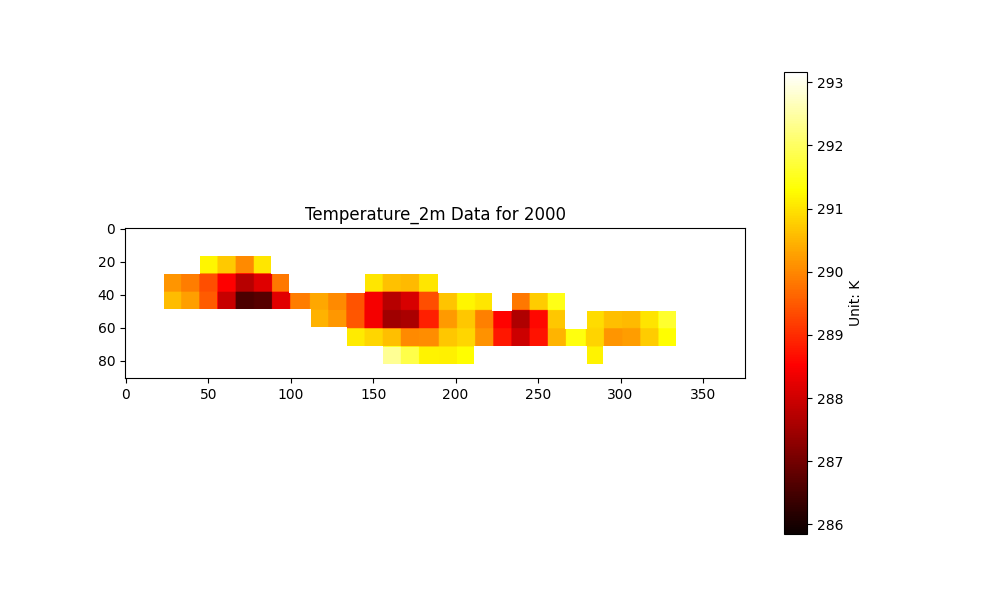
Description automatically generated**

These findings provide insights into the climatic changes and environmental conditions in Crete, contributing to better understanding and planning for climate resilience and adaptation strategies.

## Examples of Visualizations

The following examples illustrate the types of visualizations created during this project. Each visualization type is explained, along with its significance and interpretation.

1. **Static Plots**:
   * Static plots were generated for each environmental parameter, providing clear snapshots of data for specific years.

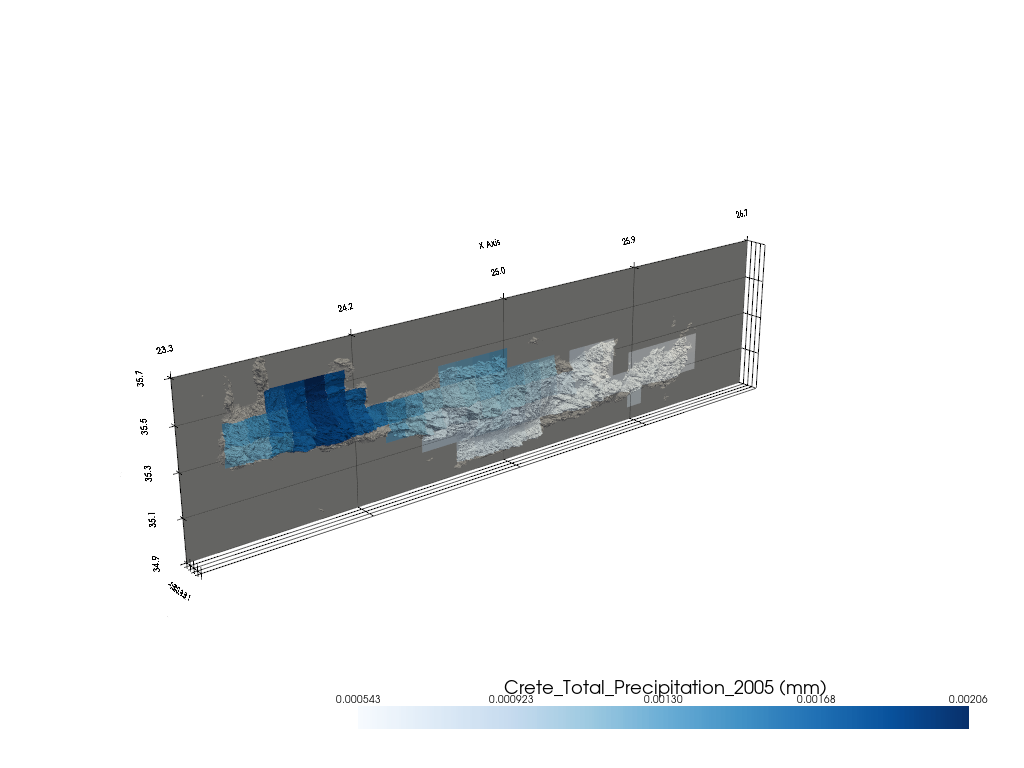


1. **Simple Line Graphs**:
   * Line graphs depicted temporal trends in numerical data, helping to identify long-term changes and patterns.

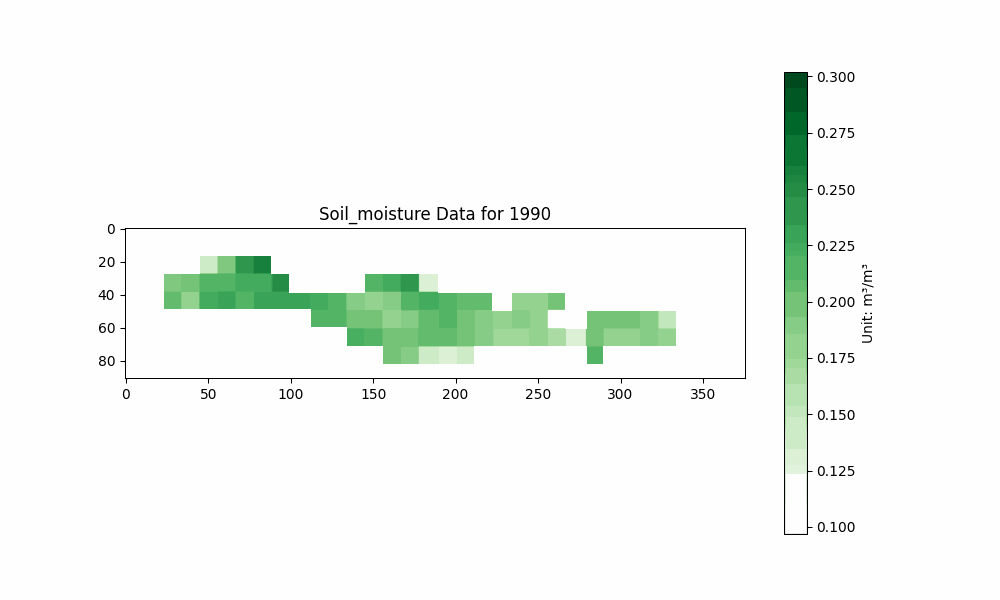
A graph with purple lines and numbers

Description automatically generated

1. **3D Models**:
   * 3D models overlaid environmental data on the digital elevation model (DEM) of Crete, offering a detailed view of spatial variations.



1. **GIF Animations**:
   * GIFs were created to animate the changes in environmental parameters over time, illustrating dynamic trends.



1. **Interactive Maps**:
   * Interactive maps allowed users to explore data in detail, with functionalities to select different bands and years.

|  |  |
| --- | --- |
| A map with several points on it  Description automatically generated | A screenshot of a map  Description automatically generated |

A map with a location pin on it

Description automatically generated

1. **Color Bars for Value Explanation**:
   * Color bars were added to explain the value ranges for each environmental parameter, enhancing the interpretability of the maps.

These visualizations collectively provide a comprehensive view of the environmental changes in Crete, leveraging different methods to capture both spatial and temporal dynamics. The integration of these visualizations into an interactive platform further enhances their utility, allowing for detailed exploration and analysis by researchers, policymakers, and the public.

# Interpretation

The visualizations developed in this thesis offer a multifaceted view of environmental data for Crete, enabling the detailed analysis of various environmental parameters over time and across different spatial dimensions. The combination of static plots, interactive maps, 3D models, and animations provides a comprehensive platform for examining temperature, precipitation, soil moisture, surface pressure, and wind patterns.

**Temperature Data:** The temperature visualizations highlight seasonal and yearly variations, revealing patterns such as the warming trend over the past few decades. These patterns are crucial for understanding climate change impacts and for planning agricultural activities, as they help identify periods of extreme temperatures and their frequency.

**Precipitation Data:** The precipitation maps and models illustrate the spatial distribution of rainfall across Crete. These visualizations help identify regions prone to drought or flooding, which is essential for water resource management and agricultural planning. The ability to track precipitation changes over time also aids in assessing the impacts of climate variability.

**Soil Moisture Data:** The soil moisture visualizations provide insights into the moisture content in different soil layers, which is vital for understanding water availability for crops. These data can help optimize irrigation strategies and predict crop yields, ultimately contributing to more efficient agricultural practices.

**Surface Pressure Data:** The surface pressure maps reveal pressure variations across Crete, which influence weather patterns and climate. Understanding these variations aids in weather forecasting and studying atmospheric conditions, which are critical for preparing for and mitigating the impacts of extreme weather events.

**Wind Data:** The wind visualizations show wind speed and direction, important for applications such as wind energy generation, weather forecasting, and agricultural planning. By visualizing wind patterns, stakeholders can better understand and harness wind resources.

The interactive maps and 3D models allow users to explore these parameters dynamically, providing a deeper understanding of how environmental factors interact and vary across different regions and time periods.

# Discussion

## Implications

The visualizations developed in this thesis have significant implications for environmental research, policy-making, and agricultural planning in Crete. By providing a comprehensive and interactive platform for visualizing environmental data, these tools help researchers and decision-makers understand complex environmental phenomena and make informed decisions.

**Climate Change Research:** The temperature and precipitation data visualizations aid in studying climate change impacts in Crete. The ability to visualize long-term trends and seasonal variations helps identify changes in climate patterns and assess their impacts on the environment and society. This knowledge is crucial for developing adaptation and mitigation strategies.

**Agricultural Planning:** The soil moisture and wind data visualizations are particularly valuable for agricultural planning. Farmers can use these tools to optimize irrigation strategies, predict crop yields, and plan for extreme weather events, leading to more efficient water use and improved crop management.

**Water Resource Management:** The precipitation and soil moisture data are crucial for water resource management. Understanding the spatial distribution of rainfall and soil moisture helps water managers make informed decisions about water allocation and storage, ensuring a sustainable water supply for agricultural, industrial, and domestic use.

## Comparison with Previous Research

Compared to previous research on environmental data visualization, this thesis offers several advancements. While earlier studies primarily focused on static maps and graphs, this work integrates multiple visualization techniques, providing a more comprehensive and interactive platform.

**Enhanced Interactivity:** Unlike traditional static maps, the interactive maps developed in this thesis allow users to select different parameters and time periods, enabling a more detailed exploration of the data. This interactivity enhances user engagement and provides deeper insights into environmental patterns.

**3D Modeling:** The inclusion of 3D models offers a new dimension to environmental data visualization. By visualizing data in three dimensions, users can better understand how environmental parameters vary across different elevations and landscapes, providing a more immersive and informative experience.

**Integrated Visualization Techniques:** This thesis combines static plots, interactive maps, 3D models, and animations into a single platform, offering a holistic view of environmental data. This integrated approach addresses the limitations of previous research, which often focused on individual visualization techniques.

## Applications

The visualizations developed in this thesis have a wide range of applications across various fields.

**Environmental Research:** Researchers can use these tools to study environmental patterns, assess climate change impacts, and develop mitigation strategies. The ability to visualize long-term trends and seasonal variations is particularly valuable for understanding complex environmental phenomena.

**Agriculture:** Farmers and agricultural planners can use the soil moisture and precipitation data visualizations to optimize irrigation strategies, predict crop yields, and plan for extreme weather events, leading to more efficient water use and improved crop management.

**Policy-Making:** Decision-makers can use these visualizations to develop informed policies for environmental protection, water resource management, and agricultural planning. The ability to visualize data dynamically and interactively helps in understanding the impacts of different policy decisions.

**Education:** These visualizations can also be used as educational tools to teach students about environmental science and data visualization techniques. The interactive maps and 3D models provide a hands-on learning experience, making complex environmental concepts more accessible and engaging.

# Conclusion

## Summary

This thesis developed a comprehensive and interactive platform for visualizing environmental data for Crete. By integrating multiple visualization techniques, including static plots, interactive maps, 3D models, and animations, this work addresses the limitations of previous research and provides a more detailed and engaging way to explore environmental data.

## Contributions

The visualizations developed in this thesis have significant implications for environmental research, agricultural planning, water resource management, and policy-making. By providing a deeper understanding of environmental patterns and trends, these tools help researchers and decision-makers make informed decisions and develop effective strategies for managing environmental challenges. The interactive and integrated nature of these visualizations represents a substantial advancement over traditional static maps and graphs.

## Future Research

Future research can build on this work by expanding the visualizations to other regions and incorporating additional environmental parameters. The development of more advanced interactive features and the integration of real-time data could further enhance the utility and impact of these visualizations. Additionally, exploring machine learning techniques to predict future environmental patterns based on historical data could provide valuable insights for long-term planning and decision-making.

# References

* meteoblue. (2023). Climate Change Crete. Retrieved from [meteoblue](https://www.meteoblue.com/en/climate-change/crete_greece_258763).
* Bank of Greece. (2011). Climate Change Impacts in Greece in the Near Future. Retrieved from [Bank of Greece](https://www.bankofgreece.gr/RelatedDocuments/WWF_Climate_change_impacts_in_Greece_in_the_near_future.pdf).
* Ekathimerini. (2023). Crete: Greece's Climate Crisis Hotspot. Retrieved from [Ekathimerini](https://www.ekathimerini.com/news/environment/1238774/crete-greeces-climate-crisis-hotspot/).
* Zhang, Z., & Kanevski, M. (2016). Environmental Data Analysis: Methods and Applications. *ResearchGate*. Retrieved from [ResearchGate](https://www.researchgate.net/publication/309633049_Environmental_Data_Analysis_Methods_and_Applications).
* Benavides, F., & Aviyente, S. (2017). Applications of Python to Evaluate Environmental Data Science Problems. *ResearchGate*. Retrieved from [ResearchGate](https://www.researchgate.net/publication/320738127_Applications_of_Python_to_evaluate_environmental_data_science_problems).
* Additional sources used for visualization techniques and tutorials (Autogis, Earth Data Science, Spatial Dev Guru, GemGIS).
* <https://developers.google.com/earth-engine/datasets/catalog/ECMWF_ERA5_LAND_DAILY_AGGR#bands>
* <https://developers.google.com/earth-engine/datasets/catalog/COPERNICUS_DEM_GLO30>