Table of Contents

List of Figure

List of Table

Dealing with the Gridded Population Datasets: A Visualization

Acknowledgement

I would like to express my sincere gratitude to my supervisors, Sophia KlauBner and Lorenz Wendt, for their ongoing guidance, support, and encouragement throughout this project. Their valuable insights and expertise have been instrumental in shaping the outcome of this work.

1. Introduction

Accurate population data is fundamental for a wide range of applications, including urban planning, disaster risk management, public health, environmental monitoring, and sustainable development. However, traditional population data, usually derived from censuses, are often limited in spatial resolution, outdated, or inaccessible at the local level. To address these limitations, Gridded Population Datasets (GPDs) have emerged as valuable alternatives. These datasets disaggregate census data into uniform grid cells using ancillary data such as satellite imagery, land use, and infrastructure, thereby offering spatially continuous and high-resolution population estimates.

The motivation for this research stems from the growing reliance on GPDs for decision-making, particularly in areas where fine-scale demographic data are unavailable or unreliable. While several gridded population products exist, such as WorldPop, LandScan, and GHS-POP, they differ in input data, modeling approaches, and spatial resolutions. These differences can lead to significant variation in population estimates across datasets, which has critical implications for their use in policy and research.

Therefore, comparing these datasets is essential to understand their relative strengths and limitations, assess their consistency, and identify the most appropriate dataset for specific applications. By evaluating and visualizing the discrepancies among GPDs in a selected area, this task facilitates a more informed and critical use of gridded population data, ultimately supporting better decision-making in geospatial and socio-environmental contexts.

2. Learning Objectives

- To gain familiarity with the understanding and visualization of geospatial data, especially raster data
- Working with programming languages for population visualization purposes

3. Approach and Challenges

This topic appeared both interesting and novel to me, making it a suitable choice to fulfill the internship requirement for my degree program. The exploration of different gridded population datasets enabled me to delve into real-world spatial challenges, and one of the most engaging aspects of the internship was the opportunity to work with programming languages such as Python, which significantly enhanced my technical skills. I started with a basic understanding, but through continuous practice and problem-solving, I was able to strengthen my coding abilities and gain confidence in working with data-driven geospatial tools.

4. Findings, Implementation

I chose to focus on the visualization aspect of this topic because it aligns closely with my academic background in GeoVisualization and Communication. Since effective communication of spatial data is a key component of my degree, working on the visual representation of gridded population datasets felt relevant and meaningful. This focus allowed me to explore how spatial patterns and data differences can be effectively illustrated, helping to bridge the gap between raw data and actionable insights. I also had the chance to experiment with different visualization tools and techniques such as Google Colab, QGIS, ArcGIS, etc., which enriched my understanding of communicating complex geospatial information more intuitively.

5. Code

Github Repository

https://github.com/Noor1621/Gridded-Population-Datasets-Visualization-

```
import ee
import geemap
import ipywidgets as widgets

# Initialize Earth Engine
ee.Initialize()

# Create output widget
output = widgets.Output()

# Dataset dictionary with GRUMP and other population datasets
datasets = {
   "CIESIN/GPWv411/GPW_UNWPP-Adjusted_Population_Density": {
      "label": "GPW v4.11 Population Density",
      "band": "unwpp-adjusted_population_density",
      "years": [2000, 2005, 2010, 2015, 2020]
   },
   "LandScan_Global": {
      "label": "LandScan Global",
```

```
        "band": "b1",
        "years": [2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021],
        "dynamic_id": "projects/sat-io/open-
datasets/ORNL/LANDSCAN_GLOBAL/landscan-global-{year}"
    },
    "WorldPop/GP/100m/pop": {
        "label": "WorldPop 100m",
        "band": "population",
        "years": [2010, 2015, 2020]
    },
    "JRC/GHSL/P2016/POP_GPW_GLOBE_V1": {
        "label": "GHS-POP",
        "band": "population",
        "years": [2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021]

    },
    "JRC/GHSL/P2023A/GHS_POP": {
        "label": "GHSL Global Population Surfaces (P2023A)",
        "band": "population_count",
        "years": list(range(1975, 2031))
    },
}

# Visualization parameters
bin_thresholds = [1, 6, 26, 51, 101, 501, 2501, 5001, 185000]
legend_labels = ['1-5', '6-25', '26-50', '51-100', '101-500', '501-2500', '2501-5000', '5001-
185000']
legend_colors = ['#ffffcc', '#fff775', '#fed976', '#feb24c', '#fd8d3c', '#f03b20', '#bd0026',
'#800026']

vis_params = {
    'min': 1,
    'max': len(legend_colors),
    'palette': legend_colors
}

def classify_image(image):
    bins = bin_thresholds
    classified = image.gt(bins[0]).And(image.lte(bins[1])).multiply(1)
    for i in range(1, len(bins) - 1):
        classified = classified.where(image.gt(bins[i]).And(image.lte(bins[i + 1])), i + 1)
    return classified

def load_dataset_by_year(dataset_id, band, year, country=None):
    try:
        if dataset_id == "LandScan_Global":
```

```
            image_id = datasets[dataset_id]["dynamic_id"].format(year=year)
            image = ee.Image(image_id).select(band)
        else:
            collection = ee.ImageCollection(dataset_id).filterDate(f'{year}-01-01',
f'{year+1}-01-01')
            if country:
                collection = collection.filterBounds(country)
            image = collection.select(band).first()
        return ee.Image(image)
    except Exception as e:
        print(f"Error loading {dataset_id} for {year}: {e}")
        return None

# Initialize map
Map = geemap.Map()
Map.add_basemap("OpenStreetMap")
Map.add_draw_control()

# Dropdown widgets
dataset_dropdown = widgets.Dropdown(
    options=[(v["label"], k) for k, v in datasets.items()],
    value="WorldPop/GP/100m/pop",
    description="Dataset:"
)

year_dropdown = widgets.Dropdown(
    options=datasets["WorldPop/GP/100m/pop"]["years"],
    value=2020,
    description="Year:"
)

# Update year options when dataset changes
def update_year_dropdown(change):
    dataset_id = change["new"]
    year_dropdown.options = datasets[dataset_id]["years"]
    year_dropdown.value = datasets[dataset_id]["years"][-1]

dataset_dropdown.observe(update_year_dropdown, names="value")

# Add legend
def add_legend():
    Map.add_legend(
        title="Population Count",
        labels=legend_labels,
        colors=legend_colors,
        position='bottomright'
```

```
    )

add_legend()

# Countries collection
countries = ee.FeatureCollection("USDOS/LSIB_SIMPLE/2017")

# Button handler
def get_population_data(b):
    with output:
        output.clear_output()
        print("Processing the selected area...")

        drawn_features = Map.draw_features
        if not drawn_features:
            print(" Please draw a polygon on the map.")
            return

        geometry = ee.Geometry(drawn_features[-1].geometry())
        Map.centerObject(geometry, zoom=9)
        Map.layers = Map.layers[:1]

        dataset_id = dataset_dropdown.value
        dataset_info = datasets[dataset_id]
        selected_year = year_dropdown.value

        if selected_year not in dataset_info["years"]:
            print(f"{dataset_info['label']} does not support year {selected_year}.")
            return

        if dataset_id == "WorldPop/GP/100m/pop":
            matching_countries = countries.filterBounds(geometry)
            num_countries = len(matching_countries.getInfo()['features'])
            print(f"Number of countries found: {num_countries}")

            country_images = []

            for country in matching_countries.getInfo()['features']:
                country_geometry = ee.Geometry(country['geometry'])
                image = load_dataset_by_year(dataset_id, dataset_info["band"], selected_year,
country=country_geometry)
                if image:
                    country_images.append(image)

            if country_images:
                combined_image = ee.ImageCollection(country_images).sum()
```

```
            clipped = combined_image.clip(geometry)
            masked = clipped.updateMask(clipped.gt(0))
            classified = classify_image(masked)

            Map.addLayer(classified, vis_params, f"{dataset_info['label']}
({selected_year})")

            stats = combined_image.reduceRegion(
                reducer=ee.Reducer.sum(),
                geometry=geometry,
                scale=1000,
                maxPixels=1e12
            ).getInfo()

            pop_total = stats.get(dataset_info["band"], None)
            if pop_total is not None:
                print(f"{dataset_info['label']} ({selected_year}) - Total population:
{pop_total:.2f}")
            else:
                print("No population data in the selected region.")
        else:
            print("No valid population data for the selected countries.")

    else:
        image = load_dataset_by_year(dataset_id, dataset_info["band"], selected_year)
        if image is None:
            print("Failed to load image.")
            return

        clipped = image.clip(geometry)
        masked = clipped.updateMask(clipped.gt(0))
        classified = classify_image(masked)

        Map.addLayer(classified, vis_params, f"{dataset_info['label']}
({selected_year})")

        stats = image.reduceRegion(
            reducer=ee.Reducer.sum(),
            geometry=geometry,
            scale=1000,
            maxPixels=1e12
        ).getInfo()

        pop_total = stats.get(dataset_info["band"], None)
        if pop_total is not None:
```

```
            print(f"{dataset_info['label']} ({selected_year}) - Total population: {pop_total:.2f}")
        else:
            print("No population data in the selected region.")

# Button widget
button = widgets.Button(description="Get Population Data")
button.on_click(get_population_data)

# Display UI
ui = widgets.VBox([dataset_dropdown, year_dropdown, button, output])
```

6. Visualization Output

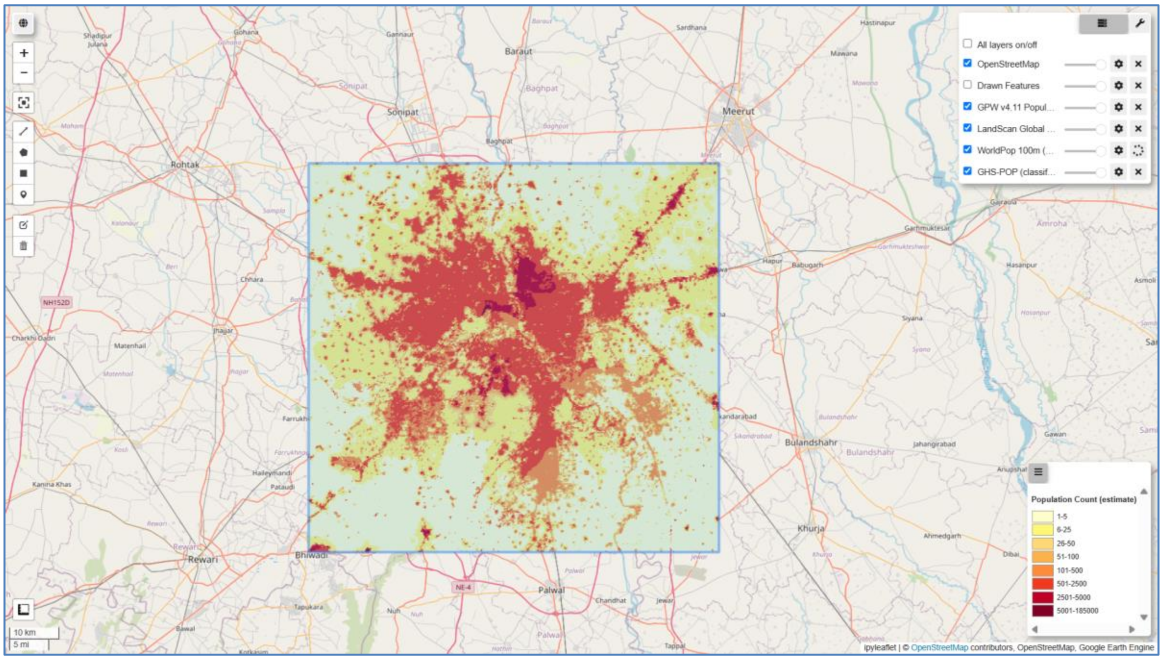

Figure 1.1
User interface of population datasets (Landscan Global, World Pop, GPWv4.11, and GHG-POP)

7. Overview and Visual Assessment of Global Population Datasets

An overview of different datasets is given below. A Random area on the map was taken as aoi and checked for the dataset visualizations as shown in Figure 2, and their attributes are given in Table 1. These datasets are similar in purpose (estimating world population), but they are not directly comparable. Each dataset differs in methodology, resolution, and underlying assumptions, approaches, and Modelling framework.

Figure 1.2
Overview of different Population Datasets

Table 1.1
Data Attribute and Results for the selected AOI

Datasets Attributes

Dataset	Year Used	Description	Data Source	Modelling Framework	Temporal Resolution	Spatial Resolution
			GEE Catalog (API)			
GPWv4.11	2020	Uses UN-adjusted census data interpolated spatially	ee.ImageCollection("CIESIN/GPWv411/GPW_Population_Count")	Top-Down	Every 5 Year	1 km
LandScan Global	2020	Distributes census data using land use, roads, and nighttime lights	ee.ImageCollection("projects/sat-io/open-datasets/ORNL/LANDSCAN_GLOBAL")	Top-Down	Annual 1 year (Average 24 hrs)	1 km
WorldPop 100m	2020	Uses machine learning with ancillary data to model population at a fine scale	ee.ImageCollection("WorldPop/GP/100m/pop")	Bottom-Up	Every 5 Year	100m
GHSL Global Population Surfaces (P2023A) (2020)	2020	fundamentally census-based, using remote sensing (built-up) as an ancillary dataset for disaggregation.	ee.ImageCollection("JRC/GHSL/P2023A/GHS_POP")	Top-down	Every 5 Year, Projections for 2025 and 2030	100 m/90m equator

Result

Dataset	Total Population Estimate (within AOI)
GPWv4.11	1196462
LandScan Global	689551
WorldPop	7161
GHSL	8223

8. Concrete Challenges

One of the major challenges I faced during the internship was that programming was not part of my academic background. As a beginner, I often struggled to understand coding syntax, structure logical workflows, and troubleshoot issues efficiently. Sometimes, resolving even small bugs took a significant amount of time and effort. However, these challenges became key learning moments. With persistence and continuous practice, I gradually improved my ability to work independently with code. This process not only enhanced my technical confidence but also sharpened my analytical thinking, especially when designing interactive geospatial environments and handling user inputs.

9. Concrete Achievements

I achieved several meaningful outcomes during the internship. I developed a Python-based environment using the Anaconda platform, where users can select their Area of Interest (AOI) and visualize population estimates from multiple gridded population datasets. These datasets were accessed and integrated through APIs, allowing for real-time interaction and comparison without the need to download or preprocess raster files. This setup provided a flexible and user-friendly way to explore population data across different sources. The project deepened my understanding of spatial data structures, improved my skills in working with API connections and Jupyter Notebooks, and allowed me to apply my knowledge of GeoVisualization in a practical, application-oriented context.

10. Future Improvements

If I had an additional month to spend on this project, I would focus on enhancing both the functionality and user experience of the interactive environment. Firstly, I would add more options for spatial and temporal filtering, allowing users to explore population data changes over time and across different geographic scales. Secondly, I would work on integrating additional gridded population datasets and including basic statistical summaries or visual comparisons, such as charts showing percentage differences, to enrich the analytical insights. I would also aim to improve the interface design to make it more intuitive and visually appealing. Additionally, I would explore deploying the tool as a web application, allowing it to be accessed and used more widely by researchers, planners, and decision-makers.

11. Recommendations for future intern

Future interns are recommended to join, as the learning environment is professional, and the staff is supportive and approachable.

12. Internship Timeline (Feb–May 2025)

Month	**Timeframe**	**Activities**
February 2025	Week 1–2	Orientation, preliminary research, and environment setup
February	Week 3–4	Exploring datasets

2025		
March 2025	Week 1–2	Designing an interactive environment and learning basic coding functionality
March 2025	Week 3–4	Implementing comparison tools and visuals, testing outputs
April 2025	Week 1–2	Refining visualizations; addressing dataset comparability
April 2025	Week 3–4	Enhancing the interface, writing documentation, and drafting the report
May 2025	Week 1 (May 1–7)	Final testing, report completion, and Submission

References:

- ee.ImageCollection("WorldPop/GP/100m/pop") https://doi.org/10.1038/sdata.2015.45
- ee.ImageCollection("projects/sat-io/open-datasets/ORNL/LANDSCAN_GLOBAL") https://doi.org/10.48690/1529167
- ee.ImageCollection("CIESIN/GPWv411/GPW_Population_Count") https://doi.org/10.7927/H4JW8BX5
- ee.ImageCollection("JRC/GHSL/P2023A/GHS_POP") https://doi.org/10.1080/17538947.2024.2390454
- https://doi.org/10.2905/2FF68A52-5B5B-4A22-8F40-C41DA8332CFE
- https://data.humdata.org/
- https://landscan.ornl.gov/
- https://developers.google.com/earth-engine/datasets