Analyzing Aadhaar Service Demand and Infrastructure Stress Patterns in India

Authors: Shreyas P Kulkarni, Shubham Bhat, Shrikar Desai, Anish A Kunder
2nd year B.Tech students at PES University

Abstract

Aadhaar is a critical digital identity infrastructure in India, supporting a wide range of administrative and welfare services. Analyzing enrolment and update activity is essential for understanding service demand and infrastructure requirements. This study examines UIDAI-provided Aadhaar enrolment, demographic update, and biometric update datasets to identify spatial patterns and demand characteristics across India.

The analysis adopts ratio-based indicators and spatial disaggregation to move beyond raw activity counts. State-level analysis captures enrolment saturation, update dominance, and regional variation, while district-level analysis focused on Karnataka highlights localized infrastructure stress. The findings reveal substantial variation in Aadhaar service demand, with regions at different stages of enrolment expansion and update-driven maturity. These insights are relevant for administrative planning and optimization of Aadhaar service delivery infrastructure.

1. PROBLEM STATEMENT AND APPROACH

1.1 PROBLEM STATEMENT:

- Aadhaar has evolved into a foundational digital identity system in India, supporting service delivery across governance, welfare, and financial inclusion. As Aadhaar adoption matures, service demand increasingly shifts from first-time enrolments to update-driven interactions, including demographic corrections and biometric updates. Understanding how this demand varies across regions is critical for ensuring efficient service delivery, infrastructure adequacy, and administrative planning.

- However, raw counts of enrolments or updates alone provide limited insight into underlying demand characteristics. They fail to distinguish between expansion-driven activity and maintenance-driven usage, and they mask regional differences in infrastructure stress. There is a need for analytical indicators that capture not only the scale of Aadhaar activity but also its intensity, composition, and spatial distribution. Identifying regions with disproportionately high repeat demand or hardware-intensive update activity can support targeted resource allocation and system improvements.

1.2 APPROACH:

- To address this problem, the study adopts a ratio-based and spatially disaggregated analytical approach using UIDAI-provided Aadhaar enrolment, demographic update, and biometric update datasets.

- First, age-wise enrolment and update counts are aggregated to compute true activity volumes. State and district names are normalized to ensure administrative consistency across datasets. The analysis then constructs interpretable indicators designed to reflect different dimensions of Aadhaar service demand, including update pressure to capture repeat usage intensity and biometric infrastructure intensity to reflect hardware-dependent operational load.

- The analysis proceeds in two stages. State-level analysis provides a macro perspective on enrolment saturation, update dominance, and regional variation across India, including statistical identification of outlier states exhibiting exceptionally high or low update pressure. District-level analysis, focused on Karnataka, examines localized patterns of infrastructure stress by combining intensity-based metrics with absolute activity volumes.

- By integrating ratio-based indicators with spatial visualization, the approach enables differentiation between scale-driven demand and intensity-driven infrastructure stress. This framework supports clearer interpretation of Aadhaar service usage patterns and offers practical insights relevant to administrative planning, capacity provisioning, and optimization of Aadhaar service delivery infrastructure.

2. DATASETS USED

2.1 AADHAAR DATASETS

This study utilizes Aadhaar enrolment and update datasets released by the Unique Identification Authority of India (UIDAI). The datasets used are described below.

- Aadhaar Enrolment Dataset.
- Aadhaar Demographic Update Dataset.
- Aadhaar Biometric Update Dataset.

2.2 SPATIAL BOUNDARY DATASETS

Publicly available spatial boundary files for geographic analysis.

- State-Level Spatial Boundary Data (India). (mapIndia.json)
- District-Level Spatial Boundary Data (Karnataka). (mapKarnataka.json)

3. METHODOLOGY

The analysis follows a structured workflow designed to ensure accurate aggregation of Aadhaar activity, administrative consistency across datasets, and interpretability of derived indicators. The methodology comprises data preprocessing, aggregation, metric construction, and spatial analysis.
The complete analysis workflow and code implementation are provided in Appendix B. Notebook also on github.

3.1 DATA PREPROCESSING AND AGGREGATION

The Aadhaar enrolment, demographic update, and biometric update datasets are provided as age-wise counts at varying geographic resolutions. To compute true activity volumes, all datasets are aggregated by summing age-bucketed counts. This approach ensures that enrolment and update totals accurately reflect the number of Aadhaar events.

State and district names are standardized to address inconsistencies arising from spelling variations, formatting differences, and administrative renaming. Text normalization is applied to harmonize case, spacing, and special characters. Fuzzy string-matching techniques are then used to align administrative names across datasets and spatial boundary files, ensuring consistent joins for aggregation and visualization.

3.2 CONSTRUCTION OF ANALYTICAL METRICS

To move beyond raw activity counts, the analysis constructs a focused set of ratio-based indicators designed to capture repeat service demand, update composition, and infrastructure-related load. Metrics are intentionally limited to avoid redundancy and ensure interpretability.

- **Update Pressure** is computed as the ratio of total Aadhaar updates (demographic and biometric combined) to new enrolments. This metric captures the intensity of repeat service demand relative to first-time registrations and serves as the primary indicator of operational load.
- **Biometric Intensity** (state-level) is calculated as the ratio of biometric updates to enrolments, reflecting the prevalence of biometric maintenance activity relative to new registrations.
- At the district level, **Biometric Infrastructure Intensity** is computed as the proportion of biometric updates within total update activity. This indicator reflects the degree to which Aadhaar service demand is hardware-dependent and operationally intensive.
- To provide scale context, **Total Aadhaar Activity** is derived as the sum of enrolments and updates, allowing differentiation between intensity-driven stress and volume-driven demand.

- In addition, **Demographic Share** and **Biometric Share** are computed as complementary proportions within total update activity to describe the composition of updates. These measures are used for descriptive interpretation rather than as independent analytical indicators.
- Statistical outlier detection is applied at the state level to identify regions exhibiting exceptionally high or low update pressure relative to the national distribution. Outlier analysis is used to highlight differing stages of Aadhaar adoption rather than as a standalone metric.

Redundant or mathematically reciprocal metrics are intentionally excluded to avoid duplication and maintain analytical clarity.

3.3 State-Level and District-Level Analysis

The analysis is conducted at two spatial scales. State-level analysis provides a macro perspective on enrolment saturation, update dominance, and regional variation across India. Statistical techniques are applied to identify states exhibiting exceptionally high or low update pressure relative to the national distribution.

District-level analysis focuses on Karnataka to examine localized patterns of infrastructure stress. At this finer resolution, ratio-based indicators are interpreted alongside absolute activity volumes to distinguish between scale-driven demand and intensity-driven operational pressure. District-level outlier analysis is not emphasized due to the absence of extreme deviations within the state distribution

3.4 Spatial Visualization

Spatial analysis is performed using GeoJSON boundary files for India and Karnataka. Choropleth maps are used to visualize geographic variation in enrolment volumes, update pressure, and biometric infrastructure intensity. Visualization parameters are chosen to ensure interpretability in both color and grayscale formats, supporting clarity in print and digital media.

4. Data Analysis and Visualisation

State-wise distributions of enrolments and update volumes are provided in Appendix A for reference.

4.1 Update Pressure

Definition:

Update Pressure is defined as the ratio of total Aadhaar updates (demographic and biometric combined) to new enrolments. It captures the intensity of repeat Aadhaar service demand relative to first-time registrations and serves as a proxy for sustained operational load on Aadhaar infrastructure.

State-level Distribution:

The state-wise distribution of update pressure reveals substantial regional variation across India. Several states and union territories exhibit markedly higher update pressure, indicating Aadhaar usage dominated by repeat update activity rather than new enrolments. This pattern is especially pronounced in administratively mature or highly urbanized regions, where enrolment volumes have stabilized and service demand is driven primarily by maintenance and correction activities.

In contrast, a number of states display comparatively low update pressure, reflecting enrolment-driven Aadhaar activity and ongoing expansion of first-time registrations. These regions are likely at earlier stages of Aadhaar adoption, where infrastructure demand is shaped more by enrolment outreach than by sustained update load.

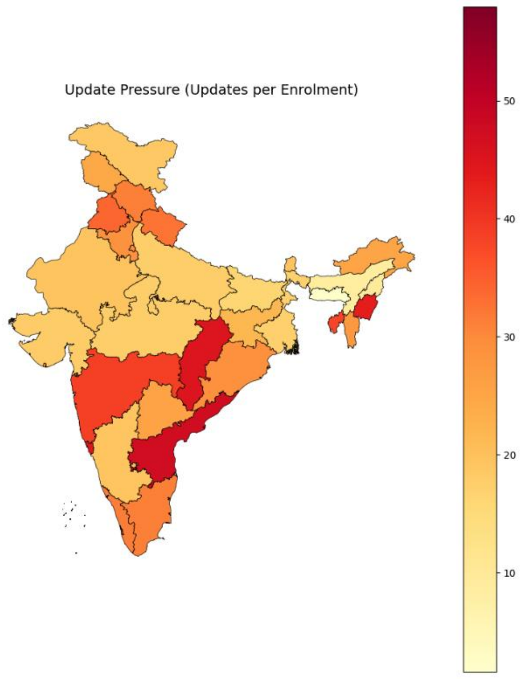


Figure 1.a

Overall, the state-level map demonstrates that absolute enrolment or update volumes alone are insufficient to characterize service demand, as regions with similar activity scale can exhibit very different update pressure profiles.

State-level Outlier Analysis:

State-wise update pressure exhibits a small number of notable outliers. **Chandigarh** (≈57.96) and the **Andaman and Nicobar Islands** (≈54.68) show exceptionally high update pressure, reflecting saturated enrolment and maintenance-dominated Aadhaar activity, amplified by small enrolment denominators. Owing to its very small geographic size, Chandigarh is not visually prominent on national-scale maps despite being a statistical extreme. In contrast, **Meghalaya** (≈1.59) exhibits unusually low update pressure, indicating enrolment-driven Aadhaar activity and continued expansion of first-time registrations.

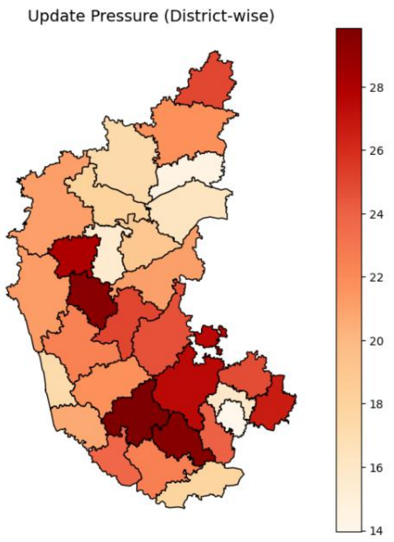


Figure 1.b

District-level Distribution (Karnataka)

District-level visualization of update pressure within Karnataka reveals pronounced intra-state variation that is not apparent at the aggregate state level. Several districts exhibit significantly higher update pressure than the state average, indicating localized concentrations of repeat Aadhaar service demand. These districts experience sustained operational load despite moderate or lower enrolment volumes.

Conversely, districts with lower update pressure correspondingly exhibit more enrolment-driven activity, suggesting continued expansion rather than maintenance-dominated usage. The coexistence of high- and low-pressure districts within the same state highlights the heterogeneity of Aadhaar service demand at finer spatial scales.

Importantly, while district-level variation is substantial, the distribution does not exhibit extreme statistical outliers. This indicates that infrastructure stress within Karnataka is distributed across multiple districts rather than concentrated in isolated locations.

Key Inferences

- Higher update pressure reflects Aadhaar usage dominated by repeat updates, indicating sustained operational demand and ongoing load on existing infrastructure.
- Lower update pressure correspondingly reflects enrolment-driven activity, suggesting regions still in expansion phases of Aadhaar adoption.
- State-level patterns mask significant intra-state heterogeneity, underscoring the importance of district-level analysis for identifying localized infrastructure stress.
- Within Karnataka, Aadhaar-related infrastructure demand is spread across multiple districts rather than concentrated in a few extreme cases. (no outliers)

4.2 Biometric Intensity and Update Composition

Definition:

Biometric Intensity is defined as the ratio of biometric updates to new enrolments at the state level. It captures the prevalence of biometric maintenance activity relative to first-time registrations and reflects the extent of hardware-dependent Aadhaar service demand. At the district level, Biometric Infrastructure Intensity is measured as the proportion of biometric updates within total update activity, indicating the relative operational burden associated with biometric processing.

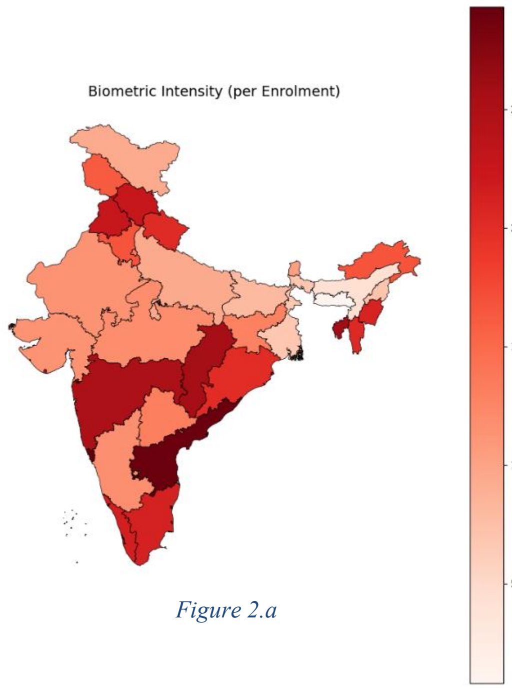


Figure 2.a

State-level Biometric Intensity

The state-wise distribution of biometric intensity reveals notable variation in the extent to which Aadhaar activity is driven by biometric maintenance rather than enrolment expansion. States exhibiting higher biometric intensity are characterized by a greater volume of biometric updates per enrolment, suggesting sustained reliance on biometric correction and re-capture processes. Such patterns are indicative of mature Aadhaar usage, where operational demand increasingly involves hardware-intensive services.

Conversely, states with lower biometric intensity display comparatively fewer biometric updates relative to enrolments, reflecting enrolment-driven
expansion or a predominance of non-biometric update activity. These regions may experience lower hardware stress per enrolment despite ongoing growth in Aadhaar coverage.
The spatial distribution demonstrates that biometric intensity does not uniformly track overall activity volumes, reinforcing the importance of separating scale effects from infrastructure-dependent demand.

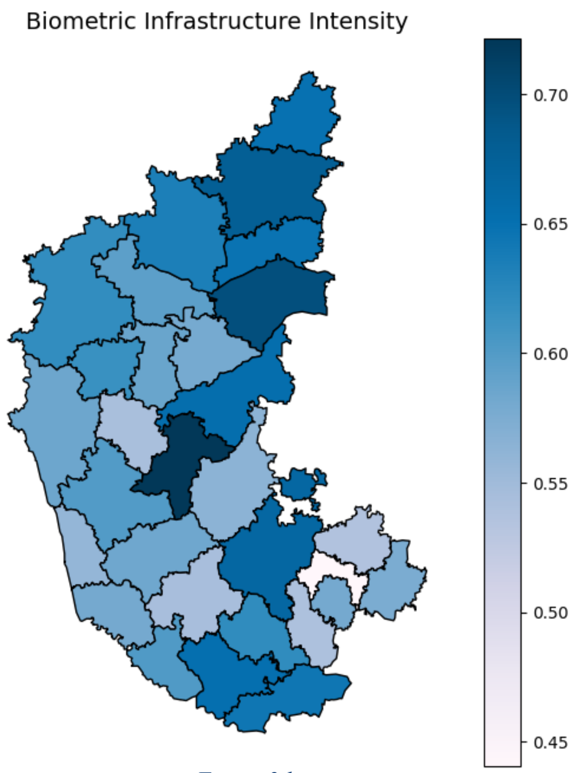


Figure 2.b

District-level Biometric Infrastructure Intensity (Karnataka)

District-level analysis of biometric infrastructure intensity within Karnataka highlights substantial intra-state variation in the composition of update activity. Several districts exhibit a high proportion of biometric updates within total updates, indicating localized concentrations of hardware-intensive service demand. These districts require sustained availability of biometric devices, trained operators, and reliable supporting infrastructure.

In contrast, districts with lower biometric infrastructure intensity are dominated by demographic updates, implying a greater share of clerical or administrative corrections. Such areas may experience lower hardware stress even when overall update volumes are moderate.

Notably, the district-level distribution does not display extreme outliers but instead shows a gradient of biometric dependence across districts. This suggests that biometric infrastructure demand within the state is spatially distributed rather than concentrated in a small number of districts.

Update Composition and Infrastructure Implications

Analysis of demographic and biometric update shares at the state level provides additional context for interpreting biometric intensity patterns. States with a higher biometric share reflect greater dependence on biometric infrastructure, while those with a higher demographic share are characterized by predominantly clerical update activity. These compositional differences help distinguish between regions where infrastructure planning must prioritize hardware provisioning versus those where staffing and administrative capacity may be more critical.

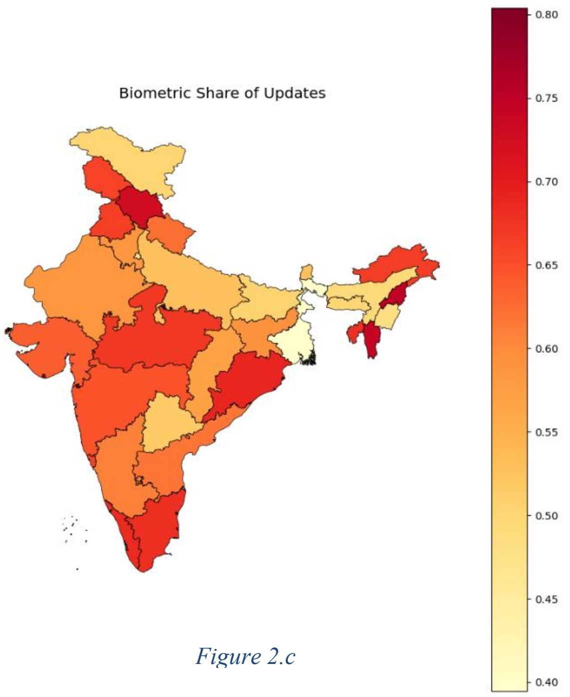


Figure 2.c

To further contextualize biometric intensity patterns, the composition of update activity is examined using the proportion of biometric updates within total updates. Regions with a higher biometric share indicate greater dependence on biometric infrastructure, while regions with a lower biometric share are dominated by clerical or demographic updates. This compositional perspective helps distinguish whether high update activity translates into hardware-intensive operational demand.

Key Inferences

- Higher biometric intensity reflects greater reliance on biometric maintenance relative to enrolments, indicating hardware-intensive Aadhaar service demand.

- Lower biometric intensity corresponds to enrolment-driven or clerical-dominated activity, suggesting comparatively lower biometric infrastructure stress.
- District-level biometric infrastructure intensity reveals heterogeneity within states that is obscured at the aggregate level.
- The composition of updates provides important context for infrastructure planning, as similar update volumes can impose very different operational requirements depending on the biometric share.

4.3 Total Aadhaar Activity (District-level Context)

Definition:

Total Aadhaar Activity is defined as the aggregate volume of Aadhaar enrolments and updates within a district. This measure provides a scale-based view of service demand and is used to contextualize intensity-based indicators such as update pressure and biometric infrastructure intensity.

Rationale for District-level Analysis:

Total Aadhaar activity is examined at the district level to provide scale context within a common administrative and policy framework. At the state level, aggregate activity volumes largely reflect population size and administrative scale and offer limited additional interpretive value. In contrast, district-level variation enables clearer differentiation between scale-driven demand and localized infrastructure stress. Accordingly, total activity is used as a contextual metric at the district level rather than as a primary state-level indicator.

District-level Distribution of Total Activity.:

Table 1

district_matched	enrolment_total	total_updates	total_activity
bengaluru urban	61273	855680	916953
belagavi	14548	307413	321961
kalaburagi	9138	198521	207659
ballari	9120	192026	201146
vijayapura	9830	170569	180399

Analysis of total Aadhaar activity across districts in Karnataka reveals substantial variation in absolute service volumes. Urban and administratively significant districts account for a large share of total Aadhaar interactions, reflecting higher population density, service accessibility, and sustained administrative demand. Table 1 lists the districts with the highest total Aadhaar activity, illustrating the concentration of overall workload within a limited number of districts.

However, districts with the highest total activity do not uniformly exhibit the highest update pressure or biometric infrastructure intensity. Several districts with moderate overall activity display elevated intensity metrics, while some high-volume districts show comparatively balanced demand profiles. This divergence highlights the importance of interpreting total activity alongside intensity-based measures rather than in isolation.

Scale versus Infrastructure Stress:

Total Aadhaar activity provides essential context for infrastructure planning but does not, by itself, indicate operational stress. High absolute workloads may be effectively managed in districts with adequate staffing and infrastructure, whereas districts with lower overall activity may experience disproportionate stress when service demand is dominated by updates or biometric operations. By jointly considering total activity with update pressure and biometric infrastructure intensity, the analysis distinguishes between scale-driven demand and intensity-driven infrastructure load.

Key Inferences:

- Total Aadhaar activity captures the absolute scale of service demand but does not directly measure infrastructure stress.
- High-volume districts do not necessarily experience the highest operational pressure, underscoring the limitations of scale-based indicators alone.
- Districts with moderate activity levels can experience elevated infrastructure stress when update demand is disproportionately high.
- Interpreting total activity alongside intensity-based metrics enables more accurate identification of districts requiring targeted capacity planning and resource allocation.

5. KEY INSIGHTS AND INFRASTRUCTURE IMPLICATIONS

5.1 KEY INSIGHTS

- Update pressure effectively distinguishes between enrolment-driven Aadhaar expansion and maintenance-driven usage across regions, revealing differing stages of Aadhaar adoption that are not evident from absolute activity volumes alone.
- State-level analysis masks substantial intra-state heterogeneity, as districts within the same state can exhibit markedly different levels of infrastructure stress and update intensity.
- High update pressure does not necessarily coincide with high total Aadhaar activity, underscoring the importance of ratio-based indicators for assessing operational load.
- Biometric intensity and update composition highlight regions where Aadhaar service demand is disproportionately hardware-dependent, indicating higher reliance on biometric devices and trained operators.
- District-level analysis within Karnataka shows that infrastructure pressure is spatially distributed across districts rather than concentrated in a small number of extreme outliers.

5.2 INFRASTRUCTURE IMPLICATIONS

The findings suggest that Aadhaar infrastructure planning should account not only for the scale of service demand but also for its intensity and composition. Regions characterized by high update pressure require sustained operational capacity to manage repeat service interactions, even when enrolment volumes are modest. In such regions, staffing levels, processing capacity, and system uptime become critical considerations.

Areas with elevated biometric intensity indicate greater dependence on hardware-intensive operations, necessitating adequate provisioning of biometric devices, regular maintenance, and skilled personnel. Conversely, regions where update activity is dominated by demographic changes may benefit more from administrative capacity enhancements and workflow optimization rather than additional biometric infrastructure.

The observed intra-state variation highlights the limitations of uniform, state-wide infrastructure strategies. Distributed planning that considers district-level demand characteristics can enable more efficient resource allocation and targeted capacity augmentation. By integrating scale-based and intensity-based indicators, administrators can better identify regions where infrastructure investments are likely to yield the greatest operational benefit.

6. CONCLUSION

This study examined Aadhaar enrolment and update patterns across India using a combination of ratio-based indicators and spatial analysis to assess service demand and infrastructure stress. By moving beyond raw activity counts, the analysis highlighted meaningful differences between enrolment-driven expansion and maintenance-dominated Aadhaar usage across regions. State-level analysis revealed substantial variation in update pressure and update composition, reflecting differing stages of Aadhaar adoption and operational demand. District-level analysis within Karnataka further demonstrated that infrastructure stress is not solely a function of absolute workload but is shaped by the intensity and nature of update activity. The coexistence of high-volume districts with moderate intensity and lower-volume districts with elevated stress underscores the importance of combining scale-based and intensity-based measures.

Overall, the findings emphasize the need for differentiated, data-driven approaches to Aadhaar infrastructure planning. Integrating enrolment volumes, update pressure, and biometric intensity can support more targeted capacity provisioning, efficient resource allocation, and improved service delivery. The analytical framework presented in this study can be extended to other regions and administrative contexts to support ongoing optimization of Aadhaar services.

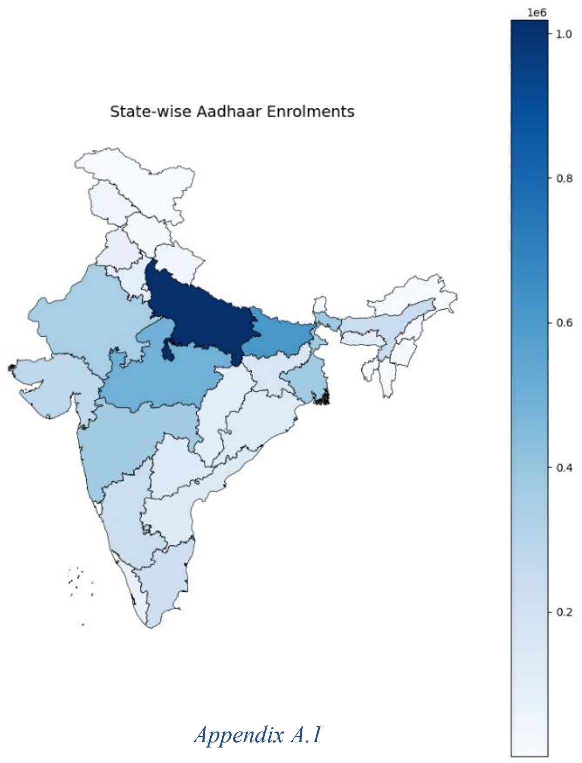
CODE AVAILABILITY

The code and notebooks used for this analysis are available at https://github.com/KadakWNL/uidai-datathon

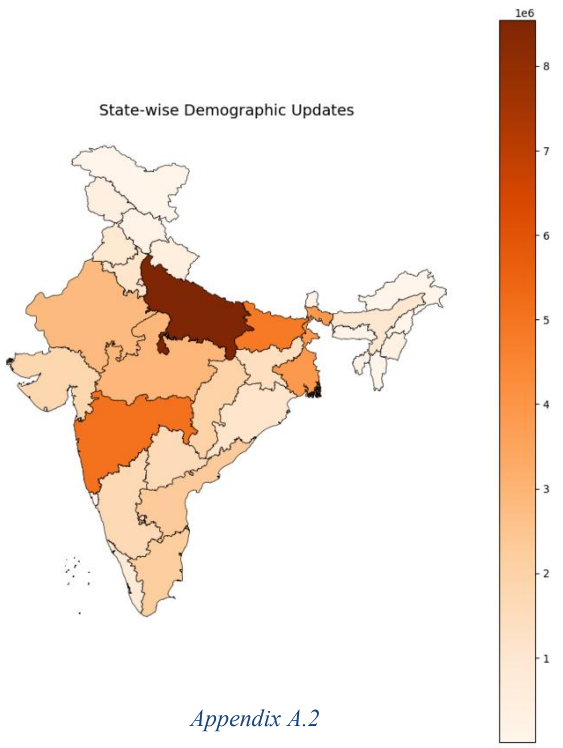
REFERENCES

1. UIDAI Datasets - https://event.data.gov.in/challenge/uidai-data-hackathon-2026/
2. India GeoJSON - https://simplemaps.com/gis/country/in
3. Karnataka GeoJSON - https://github.com/adarshbiradar/maps-geojson/blob/master/states/karnataka.json

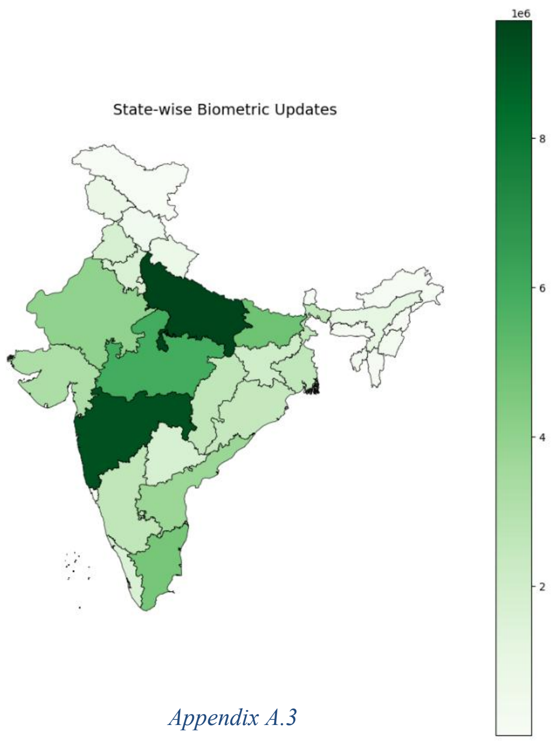
APPENDIX A – BASIC MAPS & SUPPORTING VISUALISATIONS



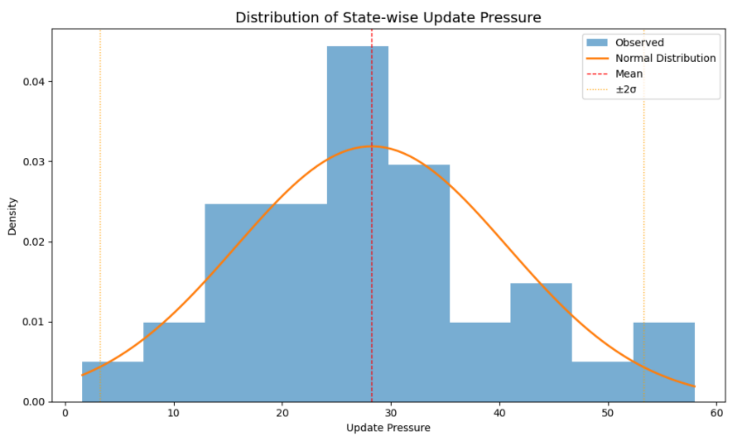
Appendix A.1



Appendix A.2



Appendix A.3



Appendix A.4

APPENDIX B – CODE

Converted code from .ipynb to text using Vertopal

- statesNB.ipynb
- districtsNB.ipynb

State-wise Aadhaar Service Demand Analysis

Objective:
This notebook performs a state-level analysis of Aadhaar enrolment and update data to identify patterns in service demand, operational load, and infrastructure stress.

Datasets Used

- Aadhaar enrolment dataset (age-wise counts)
- Aadhaar demographic update dataset
- Aadhaar biometric update dataset

The datasets are sourced from UIDAI and aggregated to the relevant spatial unit (state or district) for analysis.

Methodology Overview

- Data filtering and aggregation by geographic unit
- Normalization of region names to ensure administrative consistency
- Construction of ratio-based indicators to capture demand intensity
- Spatial visualization using choropleth maps

Redundant metrics were intentionally excluded to maintain interpretability.

```python
import pandas as pd
from pathlib import Path
import geopandas as gpd
import unicodedata
from rapidfuzz import process, fuzz

BASE_PATH = Path(".")  # notebook is at root

def load_activity_df(folder_name, activity_label):
    """Load and concatenate all CSV files from a folder."""
    dfs = []
    folder_path = BASE_PATH / folder_name

    for csv_file in folder_path.glob("*.csv"):
        df = pd.read_csv(csv_file)
        df["activity_type"] = activity_label
        df["source_file"] = csv_file.name
        dfs.append(df)
    return pd.concat(dfs, ignore_index=True)


def normalize_columns(df):
    """Normalize DataFrame column names to lowercase with underscores."""
    df.columns = df.columns.str.strip().str.lower().str.replace(" ", "_")
    return df


def drop_numeric_states(df):
    """Remove rows with purely numeric state names."""
    return df[~df["state"].str.fullmatch(r"\d+", na=False)]


def basic_state_cleanup(df):
    """Clean and standardize state name formatting."""
    df["state"] = (
        df["state"]
        .astype(str)
        .str.strip()
        .str.replace(r"\s+", " ", regex=True)
        .str.title()
    )
    return df


def apply_state_map(df, state_map):
    """Apply canonical state name mapping."""
    df["state"] = df["state"].replace(state_map)
    return df


def build_state_lookup(source_states, target_states, threshold=85):
    """Fuzzy match source states to canonical target states using rapidfuzz."""
    lookup = {}
    for s in source_states:
        match, score, _ = process.extractOne(s, target_states,
scorer=fuzz.token_sort_ratio)
        lookup[s] = match if score >= threshold else None
    return lookup


def plot_choropleth(gdf, column, title, cmap="YlOrRd", figsize=(10, 12)):
    """Create a choropleth map with consistent styling."""
    ax = gdf.plot(
        column=column,
        cmap=cmap,
        legend=True,
        figsize=figsize,
        edgecolor="black",
        linewidth=0.5
    )
    ax.set_title(title, fontsize=14)
    ax.axis("off")
    return ax
```

Loading UIDAI datasets

Here we load all UIDAI CSV files for enrolment, demographic updates, and biometric updates. Each activity type is kept in a separate dataframe to avoid mixing concepts early on.

This separation helps later when comparing how enrolments differ from update behaviour.

```
enrolment_df = load_activity_df(
    "api_data_aadhar_enrolment",
    "enrolment"
)

demographic_df = load_activity_df(
    "api_data_aadhar_demographic",
    "demographic_update"
)

biometric_df = load_activity_df(
    "api_data_aadhar_biometric",
    "biometric_update"
)
```

Data Quality Check

All datasets contain **no missing values**. The main data quality issue is **inconsistent naming** of states (e.g., Odisha/Orissa, West Bengal variants). This is addressed through canonical state mapping.

```
# normalize column names (stripping extra spaces and symbols, unicode normalization, etc)
# dropping states that is made up of only digits
enrolment_df = normalize_columns(enrolment_df)
demographic_df = normalize_columns(demographic_df)
biometric_df = normalize_columns(biometric_df)

enrolment_df = drop_numeric_states(enrolment_df)
demographic_df = drop_numeric_states(demographic_df)
biometric_df = drop_numeric_states(biometric_df)

enrolment_df = basic_state_cleanup(enrolment_df)
demographic_df = basic_state_cleanup(demographic_df)
biometric_df = basic_state_cleanup(biometric_df)
```

State Name Standardization (Systematic Approach)

Step 1: Define canonical states from GeoJSON
Step 2: Use fuzzy matching to auto-map obvious variants
Step 3: Identify what fuzzy matching missed using set difference
Step 4: Manually correct edge cases and ambiguous matches

```python
# getting state names from mapIndia.json
india_gdf_temp = gpd.read_file("mapIndia.json")
india_gdf_temp = india_gdf_temp.rename(columns={"name": "state"})
india_gdf_temp["state"] = (
    india_gdf_temp["state"]
    .astype(str)
    .str.strip()
    .str.replace(r"\s+", " ", regex=True)
    .str.title()
)

# states extracted from map
CANONICAL_STATES = {
    "Andaman And Nicobar Islands", "Andhra Pradesh", "Arunachal Pradesh", "Assam",
    "Bihar", "Chandigarh", "Chhattisgarh", "Dadra And Nagar Haveli And Daman And Diu",
    "Delhi", "Goa", "Gujarat", "Haryana", "Himachal Pradesh", "Jammu And Kashmir",
    "Jharkhand", "Karnataka", "Kerala", "Ladakh", "Lakshadweep", "Madhya Pradesh",
    "Maharashtra", "Manipur", "Meghalaya", "Mizoram", "Nagaland", "Odisha",
    "Puducherry", "Punjab", "Rajasthan", "Sikkim", "Tamil Nadu", "Telangana",
    "Tripura", "Uttar Pradesh", "Uttarakhand", "West Bengal"
}

# all unique state names from our datasets
all_states = (
    pd.concat([
        enrolment_df["state"],
        demographic_df["state"],
        biometric_df["state"]
    ])
    .unique()
)

print(f"Found {len(all_states)} unique state names in the data")
print(f"Expected {len(CANONICAL_STATES)} canonical states")
```

```
Found 58 unique state names in the data
Expected 36 canonical states
```

```python
# automating state name match using rapidfuzz by fuzzymatching
fuzzy_lookup = build_state_lookup(all_states, CANONICAL_STATES, threshold=85)

# states that couldnt be matched using fuzzymatching
unmatched_by_fuzzy = [s for s, v in fuzzy_lookup.items() if v is None]

print(f"\nFuzzy matching successfully mapped: {len([v for v in fuzzy_lookup.values() if v
is not None])} states")
print(f"Fuzzy matching FAILED on: {len(unmatched_by_fuzzy)} entries")
print(f"\nStates that need manual mapping:")
for state in sorted(unmatched_by_fuzzy):
    print(f"  - '{state}'")
fuzzy_lookup
```

```
Fuzzy matching successfully mapped: 42 states
Fuzzy matching FAILED on: 16 entries

States that need manual mapping:
  - 'Balanagar'
  - 'Dadra & Nagar Haveli'
  - 'Dadra And Nagar Haveli'
  - 'Daman & Diu'
  - 'Daman And Diu'
  - 'Darbhanga'
  - 'Jaipur'
  - 'Madanapalle'
  - 'Nagpur'
  - 'Orissa'
  - 'Pondicherry'
  - 'Puttenahalli'
  - 'Raja Annamalai Puram'
  - 'Tamilnadu'
  - 'Uttaranchal'
  - 'Westbengal'

{'Meghalaya': 'Meghalaya',
 'Karnataka': 'Karnataka',
 'Uttar Pradesh': 'Uttar Pradesh',
 'Bihar': 'Bihar',
 'Maharashtra': 'Maharashtra',
 'Haryana': 'Haryana',
 'Rajasthan': 'Rajasthan',
 'Punjab': 'Punjab',
 'Delhi': 'Delhi',
 'Madhya Pradesh': 'Madhya Pradesh',
 'West Bengal': 'West Bengal',
 'Assam': 'Assam',
 'Uttarakhand': 'Uttarakhand',
 'Gujarat': 'Gujarat',
 'Andhra Pradesh': 'Andhra Pradesh',
 'Tamil Nadu': 'Tamil Nadu',
 'Chhattisgarh': 'Chhattisgarh',
 'Jharkhand': 'Jharkhand',
 'Nagaland': 'Nagaland',
 'Manipur': 'Manipur',
 'Telangana': 'Telangana',
 'Tripura': 'Tripura',
 'Mizoram': 'Mizoram',
 'Jammu And Kashmir': 'Jammu And Kashmir',
 'Chandigarh': 'Chandigarh',
 'Sikkim': 'Sikkim',
 'Odisha': 'Odisha',
 'Kerala': 'Kerala',
 'The Dadra And Nagar Haveli And Daman And Diu': 'Dadra And Nagar Haveli And Daman And
Diu',
 'Arunachal Pradesh': 'Arunachal Pradesh',
 'Himachal Pradesh': 'Himachal Pradesh',
 'Goa': 'Goa',
 'Dadra And Nagar Haveli And Daman And Diu': 'Dadra And Nagar Haveli And Daman And Diu',
 'Ladakh': 'Ladakh',
 'Andaman And Nicobar Islands': 'Andaman And Nicobar Islands',
 'Orissa': None,
 'Pondicherry': None,
 'Puducherry': 'Puducherry',
 'Lakshadweep': 'Lakshadweep',
 'Andaman & Nicobar Islands': 'Andaman And Nicobar Islands',
 'Dadra & Nagar Haveli': None,
 'Dadra And Nagar Haveli': None,
 'Daman And Diu': None,
 'Jammu & Kashmir': 'Jammu And Kashmir',
 'Daman & Diu': None,
 'West Bangal': 'West Bengal',
 'Westbengal': None,
 'Chhatisgarh': 'Chhattisgarh',
 'West Bengli': 'West Bengal',
 'Darbhanga': None,
 'Puttenahalli': None,
 'Uttaranchal': None,
 'Balanagar': None,
 'Jaipur': None,
 'Madanapalle': None,
 'Nagpur': None,
 'Raja Annamalai Puram': None,
 'Tamilnadu': None}
```

Manual Corrections for Edge Cases

Fuzzy matching can't handle:

- **Legacy names** (Orissa → Odisha, Uttaranchal → Uttarakhand)
- **Merged territories** (Dadra & Daman merged post-2020)
- **Extreme spelling variants** (WESTBENGAL, West Bengli, etc.)
- **City leakage** (city names appearing in state column)

So we manually map these edge cases below:

```
# fuzzy matching map for state names
STATE_CANONICAL_MAP = fuzzy_lookup.copy()

# manual corrections for what fuzzy matching missed or got wrong
MANUAL_CORRECTIONS = {
    # Legacy state names
    "Orissa": "Odisha",
    "Uttaranchal": "Uttarakhand",
    "Pondicherry": "Puducherry",

    # merged territories
    "Dadra & Nagar Haveli": "Dadra And Nagar Haveli And Daman And Diu",
    "Dadra and Nagar Haveli": "Dadra And Nagar Haveli And Daman And Diu",
    "Dadra And Nagar Haveli": "Dadra And Nagar Haveli And Daman And Diu",
    "Daman & Diu": "Dadra And Nagar Haveli And Daman And Diu",
    "Daman and Diu": "Dadra And Nagar Haveli And Daman And Diu",
    "Daman And Diu": "Dadra And Nagar Haveli And Daman And Diu",


    # City names found in state column (data quality issue)
    "Darbhanga": "Bihar",
    "Balanagar": "Telangana",
    "Jaipur": "Rajasthan",
    "Madanapalle": "Andhra Pradesh",
    "Puttenahalli": "Karnataka",
    "Nagpur": "Maharashtra",
    "Raja Annamalai Puram": "Tamil Nadu"
}

STATE_CANONICAL_MAP.update(MANUAL_CORRECTIONS)

# combined mapping
enrolment_df = apply_state_map(enrolment_df, STATE_CANONICAL_MAP)
demographic_df = apply_state_map(demographic_df, STATE_CANONICAL_MAP)
biometric_df = apply_state_map(biometric_df, STATE_CANONICAL_MAP)

# check if any states still don't match canonical set
final_states = set(pd.concat([
    enrolment_df["state"],
    demographic_df["state"],
    biometric_df["state"]
]).unique())

unmatched_final = final_states - CANONICAL_STATES
if len(unmatched_final) > 0:
    print(f"{len(unmatched_final)} states still not matching canonical set:")
    print(unmatched_final)
else:
    print("All states successfully mapped to canonical names!")
```

```
1 states still not matching canonical set:
{None}
```

Geospatial Setup & Metric Calculation

Loading the India GeoJSON and aligning state names with our cleaned data.

Key metrics we'll calculate:

- **Enrolments**: New Aadhaar cards created
- **Demographic updates**: Address/name/DOB changes
- **Biometric updates**: Fingerprint/iris corrections

- **Update pressure**: How many updates happen per new enrolment (our main metric!)

```
# load India map and clean state names to match our data
india_gdf = gpd.read_file("mapIndia.json")
india_gdf = india_gdf.rename(columns={"name": "state"})
india_gdf["state"] = (
    india_gdf["state"]
    .astype(str)
    .str.strip()
    .str.replace(r"\s+", " ", regex=True)
    .str.title()
)

def strip_accents(text):
    """Remove diacritical marks (accents) from text."""
    if not isinstance(text, str):
        return text
    return "".join(
        c for c in unicodedata.normalize("NFKD", text)
        if not unicodedata.combining(c)
    )

india_gdf["state"] = india_gdf["state"].replace(STATE_CANONICAL_MAP)
india_gdf["state"] = india_gdf["state"].apply(strip_accents)

# aggregate to state level - sum all age groups (not just count rows!)
state_enrol = (
    enrolment_df.groupby("state")[["age_0_5", "age_5_17", "age_18_greater"]]
    .sum()
    .sum(axis=1)
    .rename("enrolments")
)
state_demo = (
    demographic_df.groupby("state")[["demo_age_5_17", "demo_age_17_"]]
    .sum()
    .sum(axis=1)
    .rename("demographic_updates")
)
state_bio = (
    biometric_df.groupby("state")[["bio_age_5_17", "bio_age_17_"]]
    .sum()
    .sum(axis=1)
    .rename("biometric_updates")
)

# Join them together
state_summary = (
    state_enrol.to_frame()
    .join(state_demo, how="left")
    .join(state_bio, how="left")
    .fillna(0)
)
```

```
# Calculate our key metrics
# UPDATE PRESSURE = how many updates per enrolment (main metric!)
state_summary["update_pressure"] = (
    (state_summary["demographic_updates"] + state_summary["biometric_updates"])
    / state_summary["enrolments"]
)

state_summary["biometric_intensity"] = (
    state_summary["biometric_updates"] / state_summary["enrolments"]
)

# What % of updates are demographic vs biometric?
state_summary["demographic_share"] = (
    state_summary["demographic_updates"]
    / (state_summary["demographic_updates"] + state_summary["biometric_updates"])
)

state_summary["biometric_share"] = (
    state_summary["biometric_updates"]
    / (state_summary["demographic_updates"] + state_summary["biometric_updates"])
)

# merge with map for visualization
india_summary_gdf = india_gdf.merge(
    state_summary.reset_index(),
    on="state",
    how="left"
)
```

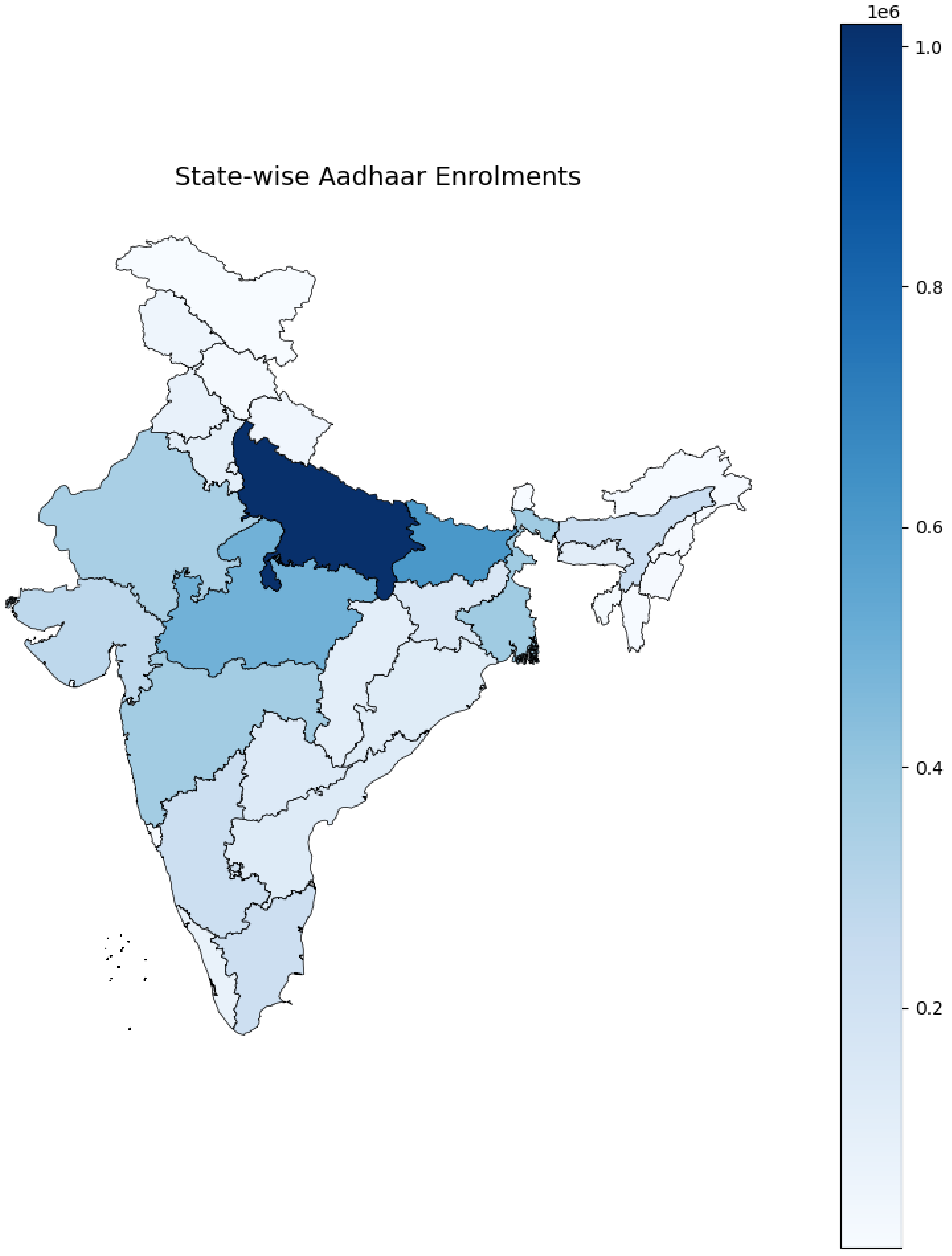
Spatial Visualization

Baseline Distribution Maps

Raw counts show where Aadhaar activity is concentrated geographically.

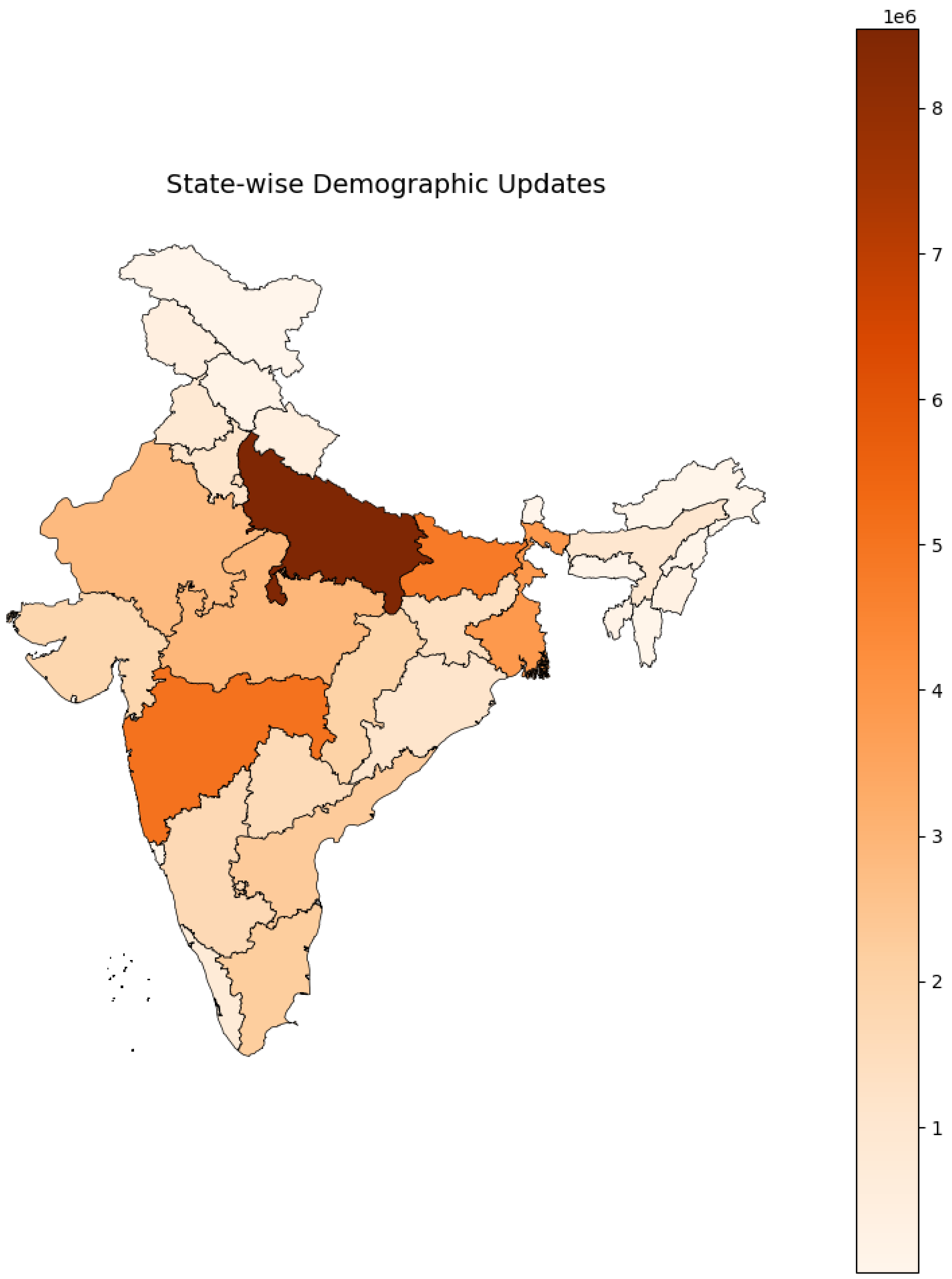
```
plot_choropleth(india_summary_gdf, "enrolments", "State-wise Aadhaar Enrolments",
"Blues")
```

```
<Axes: title={'center': 'State-wise Aadhaar Enrolments'}>
```



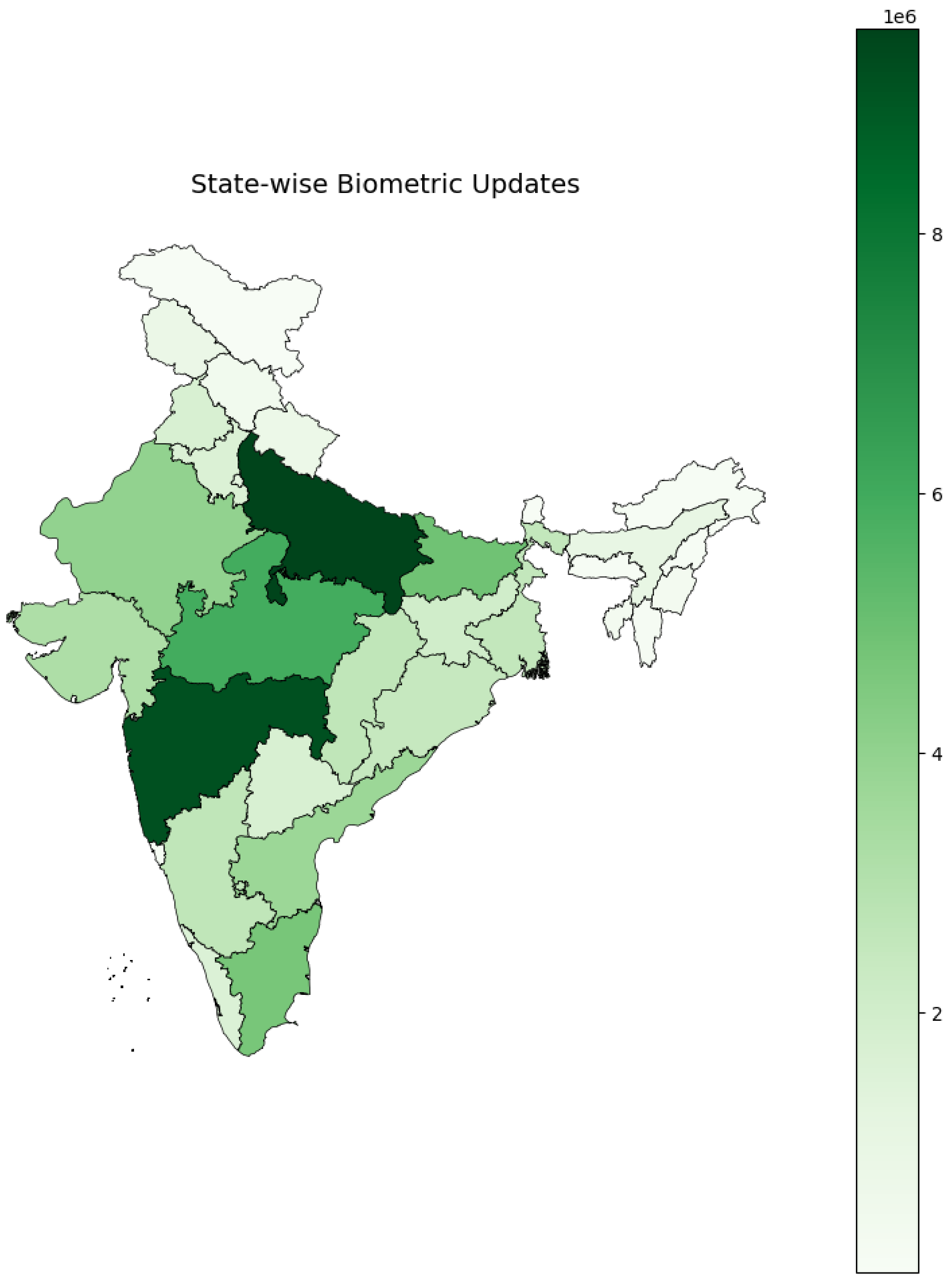
```
plot_choropleth(india_summary_gdf, "demographic_updates", "State-wise Demographic
Updates", "Oranges")
```

```
<Axes: title={'center': 'State-wise Demographic Updates'}>
```



```
plot_choropleth(india_summary_gdf, "biometric_updates", "State-wise Biometric Updates",
"Greens")
```

```
<Axes: title={'center': 'State-wise Biometric Updates'}>
```

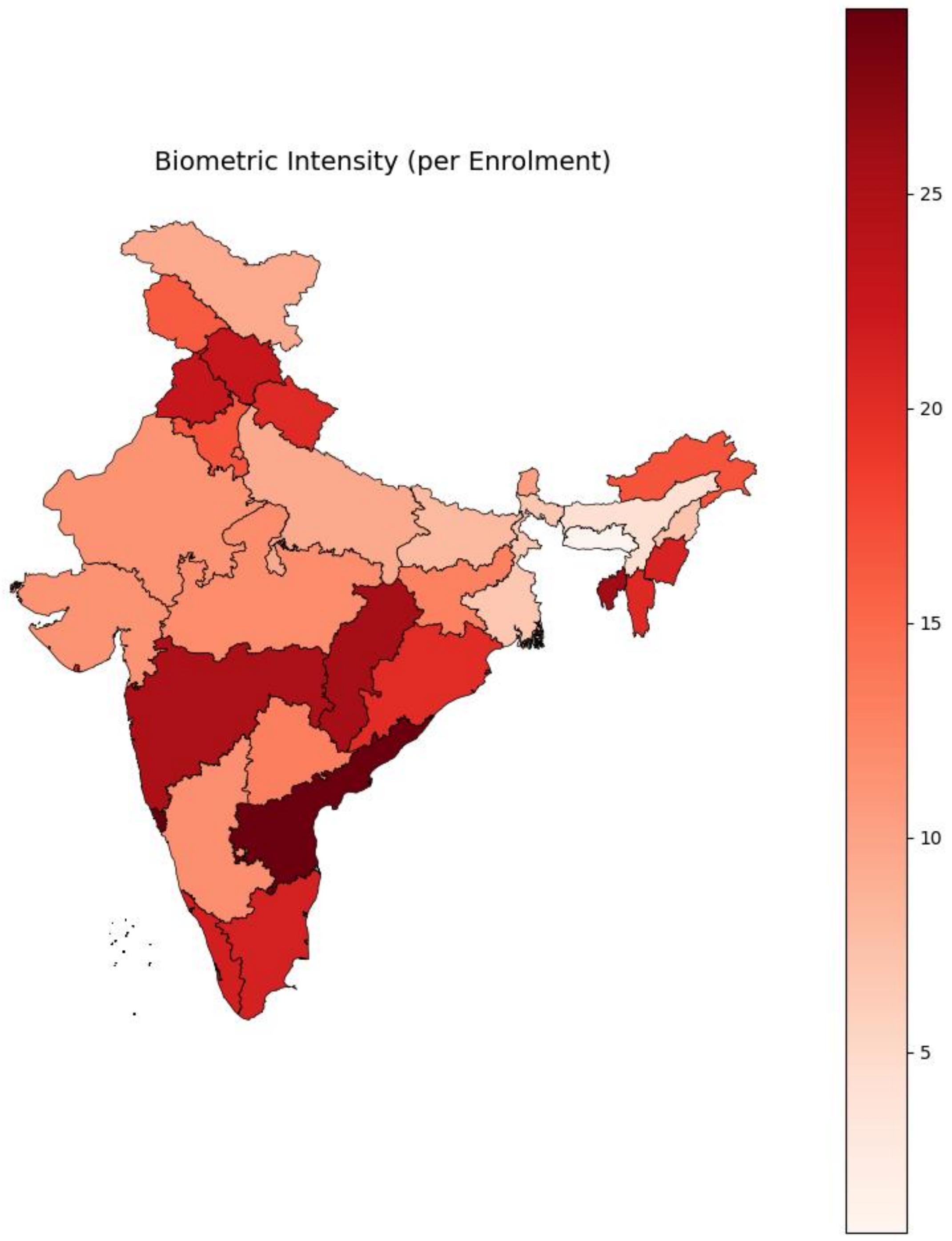


Normalized Metrics

Ratio-based indicators enable fair comparison across states by accounting for scale differences.

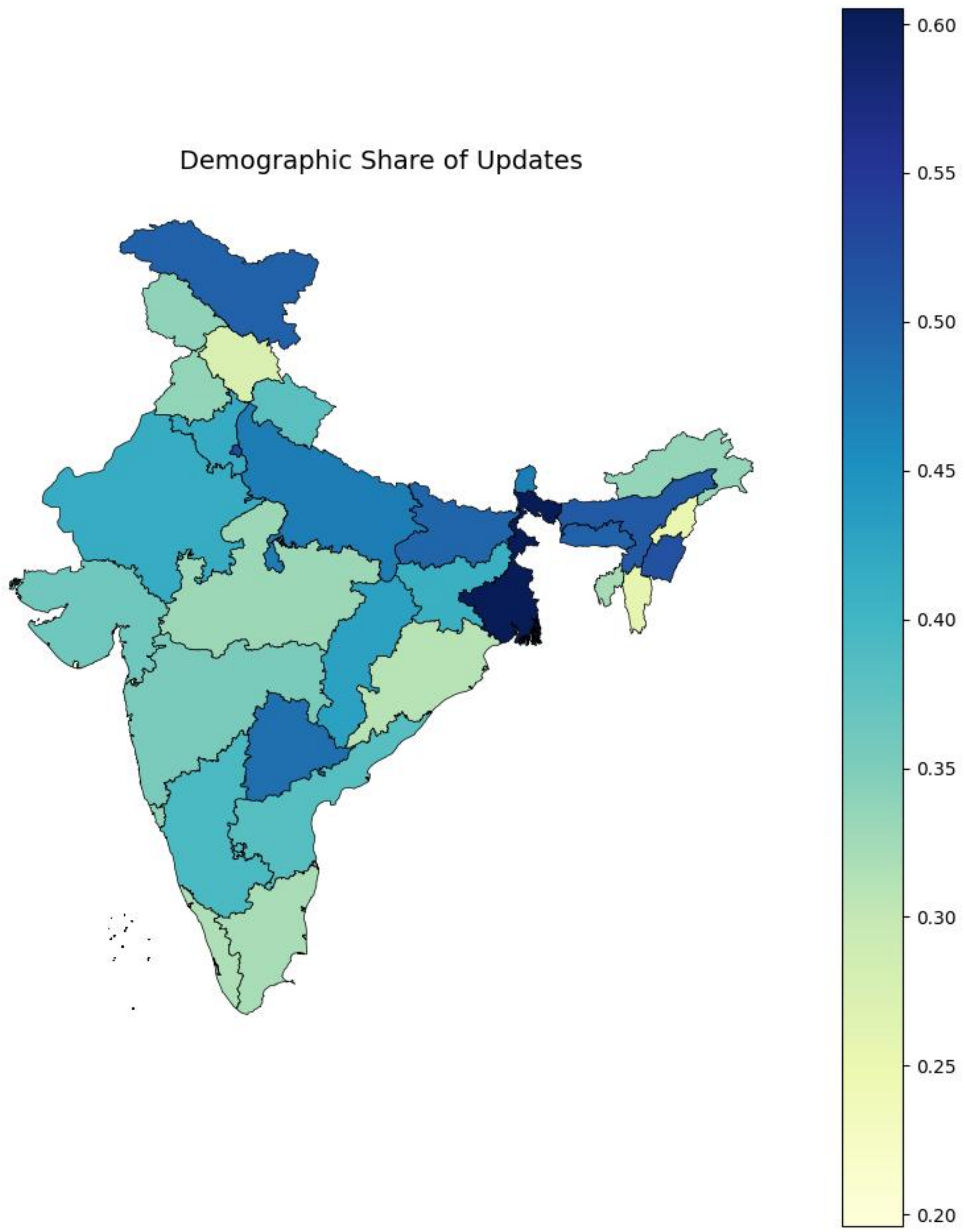
```
plot_choropleth(india_summary_gdf, "biometric_intensity", "Biometric Intensity (per
Enrolment)", "Reds")
```

```
<Axes: title={'center': 'Biometric Intensity (per Enrolment)'}>
```



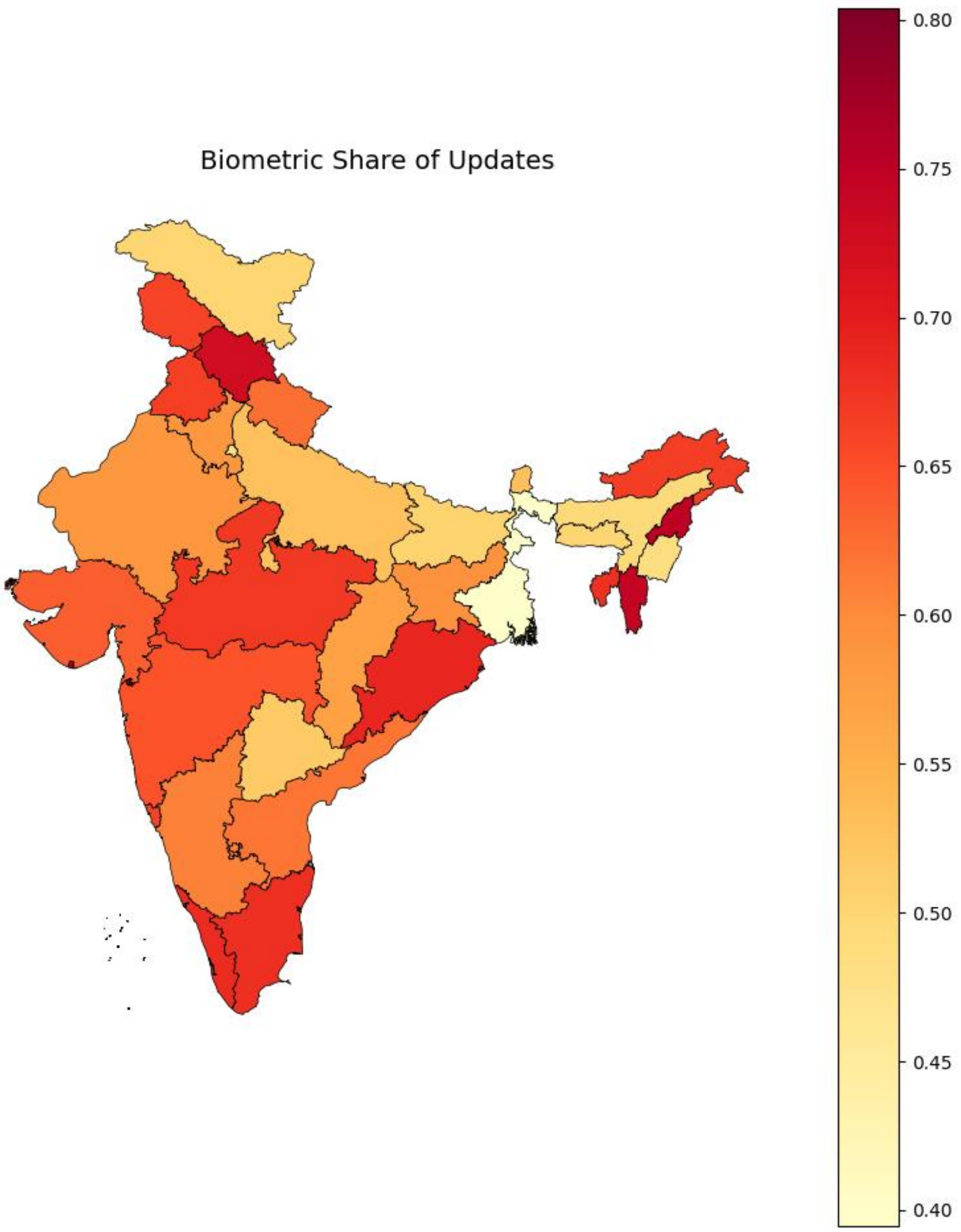
```
plot_choropleth(india_summary_gdf, "demographic_share", "Demographic Share of Updates",
"YlGnBu")
```

```
<Axes: title={'center': 'Demographic Share of Updates'}>
```



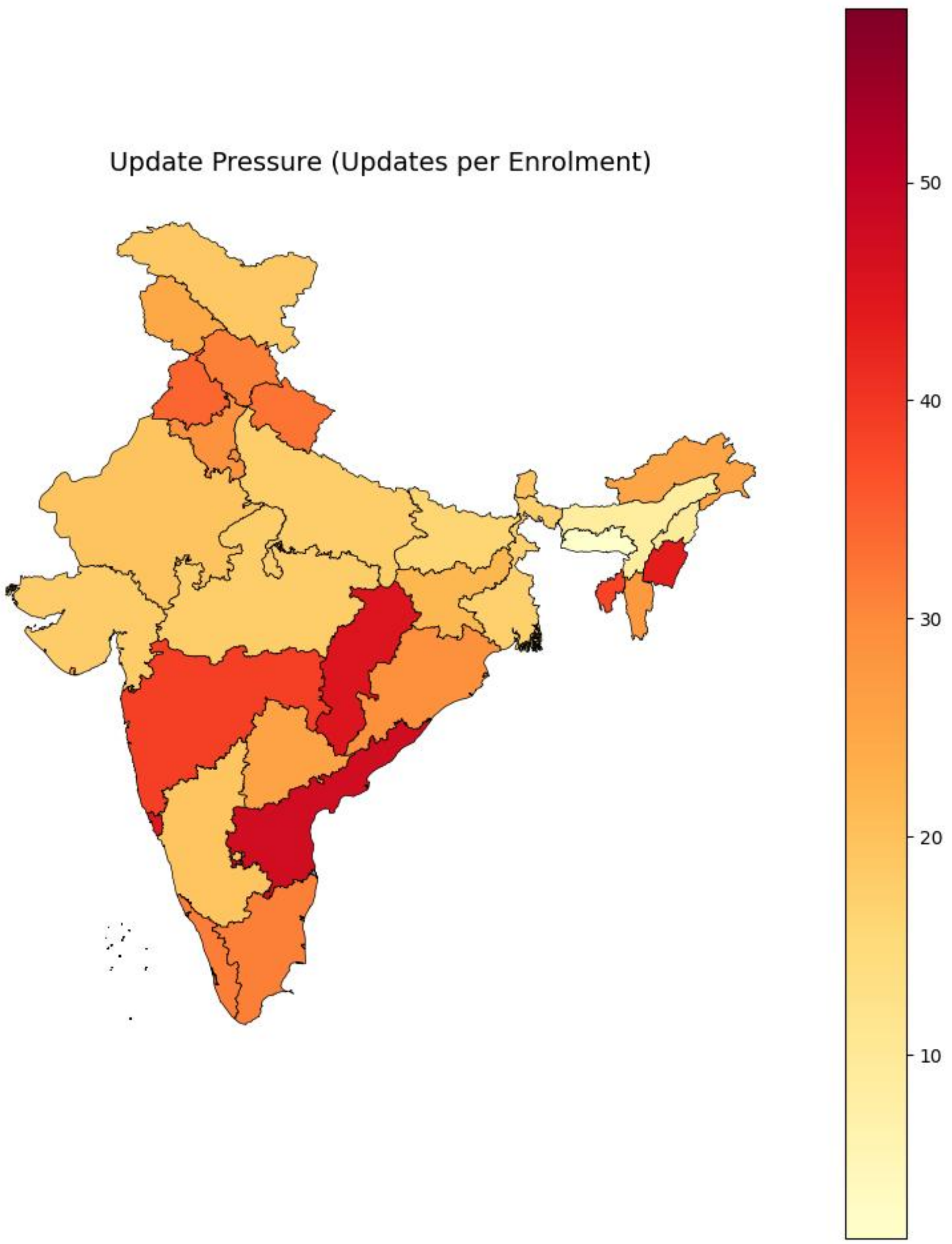
```
plot_choropleth(india_summary_gdf, "biometric_share", "Biometric Share of Updates",
"YlOrRd")
```

```
<Axes: title={'center': 'Biometric Share of Updates'}>
```



```
plot_choropleth(india_summary_gdf, "update_pressure", "Update Pressure (Updates per
Enrolment)", "YlOrRd")
```

```
<Axes: title={'center': 'Update Pressure (Updates per Enrolment)'}>
```



Outlier Detection

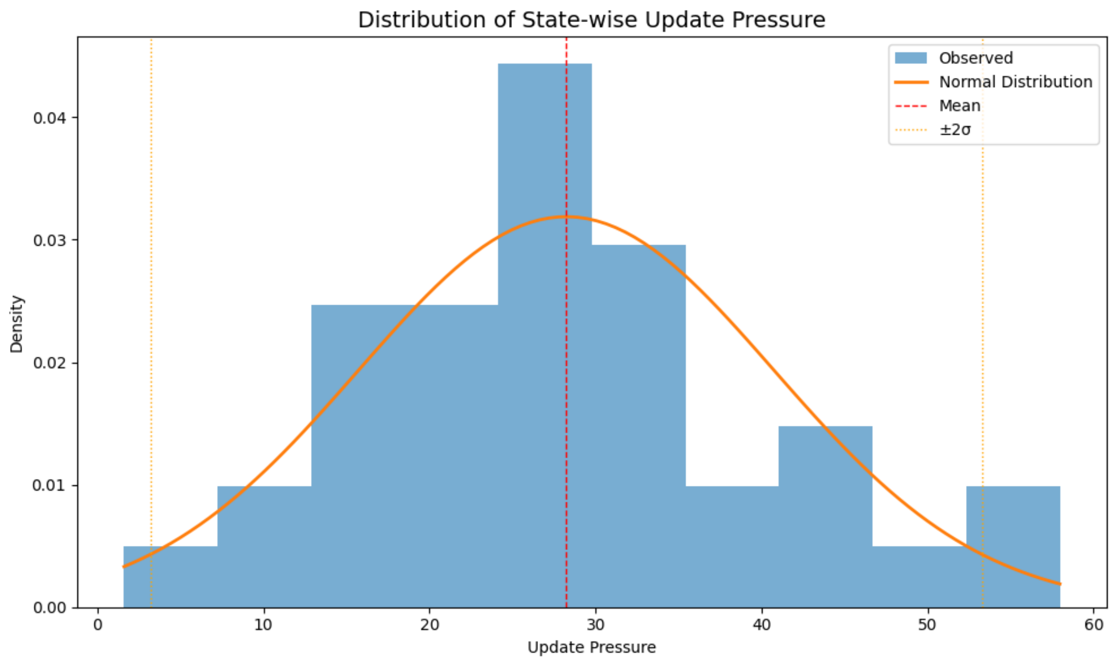
Using z-scores to find states that behave very differently from the national pattern.
States with |z-score| > 2 are considered outliers (roughly top/bottom 5%).

```
# outlier states (z-score > 2)
state_summary["pressure_zscore"] = (
    (state_summary["update_pressure"] - state_summary["update_pressure"].mean())
    / state_summary["update_pressure"].std()
)
```

```
outliers = state_summary.loc[
    state_summary["pressure_zscore"].abs() > 2,
    ["update_pressure", "pressure_zscore"]
].sort_values("pressure_zscore", ascending=False)

outliers
```

```
                              update_pressure  pressure_zscore
state
Chandigarh                          57.966581         2.375915
Andaman And Nicobar Islands         54.684932         2.113691
Meghalaya                            1.594264        -2.128572
```

```
import matplotlib.pyplot as plt
import numpy as np

# normal curve analysis for update pressure
pressure = state_summary["update_pressure"]
mu, sigma = pressure.mean(), pressure.std()

plt.figure(figsize=(10, 6))
plt.hist(pressure, bins=10, density=True, alpha=0.6, label="Observed")

x = np.linspace(pressure.min(), pressure.max(), 100)
normal_curve = (1 / (sigma * np.sqrt(2 * np.pi))) * np.exp(-0.5 * ((x - mu) / sigma) **
2)
plt.plot(x, normal_curve, linewidth=2, label="Normal Distribution")

plt.axvline(mu, color="red", linestyle="--", linewidth=1, label="Mean")
plt.axvline(mu + 2*sigma, color="orange", linestyle=":", linewidth=1, label="±2σ")
plt.axvline(mu - 2*sigma, color="orange", linestyle=":", linewidth=1)

plt.title("Distribution of State-wise Update Pressure", fontsize=14)
plt.xlabel("Update Pressure")
plt.ylabel("Density")
plt.legend()
plt.tight_layout()
plt.show()
```



District-wise Aadhaar Infrastructure Stress Analysis (Karnataka)

Objective:
This notebook performs a district-level analysis of Aadhaar enrolment and update data to identify patterns in service demand, operational load, and infrastructure stress.

Datasets Used

- Aadhaar enrolment dataset (age-wise counts)
- Aadhaar demographic update dataset
- Aadhaar biometric update dataset

The datasets are sourced from UIDAI and aggregated to the relevant spatial unit (state or district) for analysis.

Methodology Overview

- Data filtering and aggregation by geographic unit
- Normalization of region names to ensure administrative consistency
- Construction of ratio-based indicators to capture demand intensity
- Spatial visualization using choropleth maps

Redundant metrics were intentionally excluded to maintain interpretability.

```
import pandas as pd
from pathlib import Path
import geopandas as gpd
import unicodedata
import re
```

```
from rapidfuzz import process, fuzz

BASE_PATH = Path(".")

def load_activity_df(folder_name, activity_label):
    """Load and concatenate all CSV files from a folder."""
    dfs = []
    folder_path = BASE_PATH / folder_name

    for csv_file in folder_path.glob("*.csv"):
        df = pd.read_csv(csv_file)
        df["activity_type"] = activity_label
        df["source_file"] = csv_file.name
        dfs.append(df)

    return pd.concat(dfs, ignore_index=True)


def normalize_text(s):
    """Normalize district names by removing accents, lowercasing, and cleaning."""
    if pd.isna(s):
        return None
    s = unicodedata.normalize("NFKD", s)
    s = s.encode("ascii", "ignore").decode("utf-8")
    s = s.lower()
    s = re.sub(r"[^a-z\s]", " ", s)
    s = s.replace("&", "and")
    return " ".join(s.split())


def build_district_lookup(source_districts, target_districts, threshold=85):
    """Fuzzy match source districts to canonical target districts."""
    lookup = {}
    for d in source_districts:
        match, score, _ = process.extractOne(d, target_districts,
scorer=fuzz.token_sort_ratio)
        lookup[d] = match if score >= threshold else None
    return lookup


def plot_choropleth(gdf, column, title, cmap="OrRd", figsize=(8, 8)):
    """Create a choropleth map with consistent styling."""
    ax = gdf.plot(
        column=column,
        cmap=cmap,
        legend=True,
        figsize=figsize,
        edgecolor="black"
    )
    ax.set_title(title, fontsize=14)
    ax.axis("off")
    return ax
```

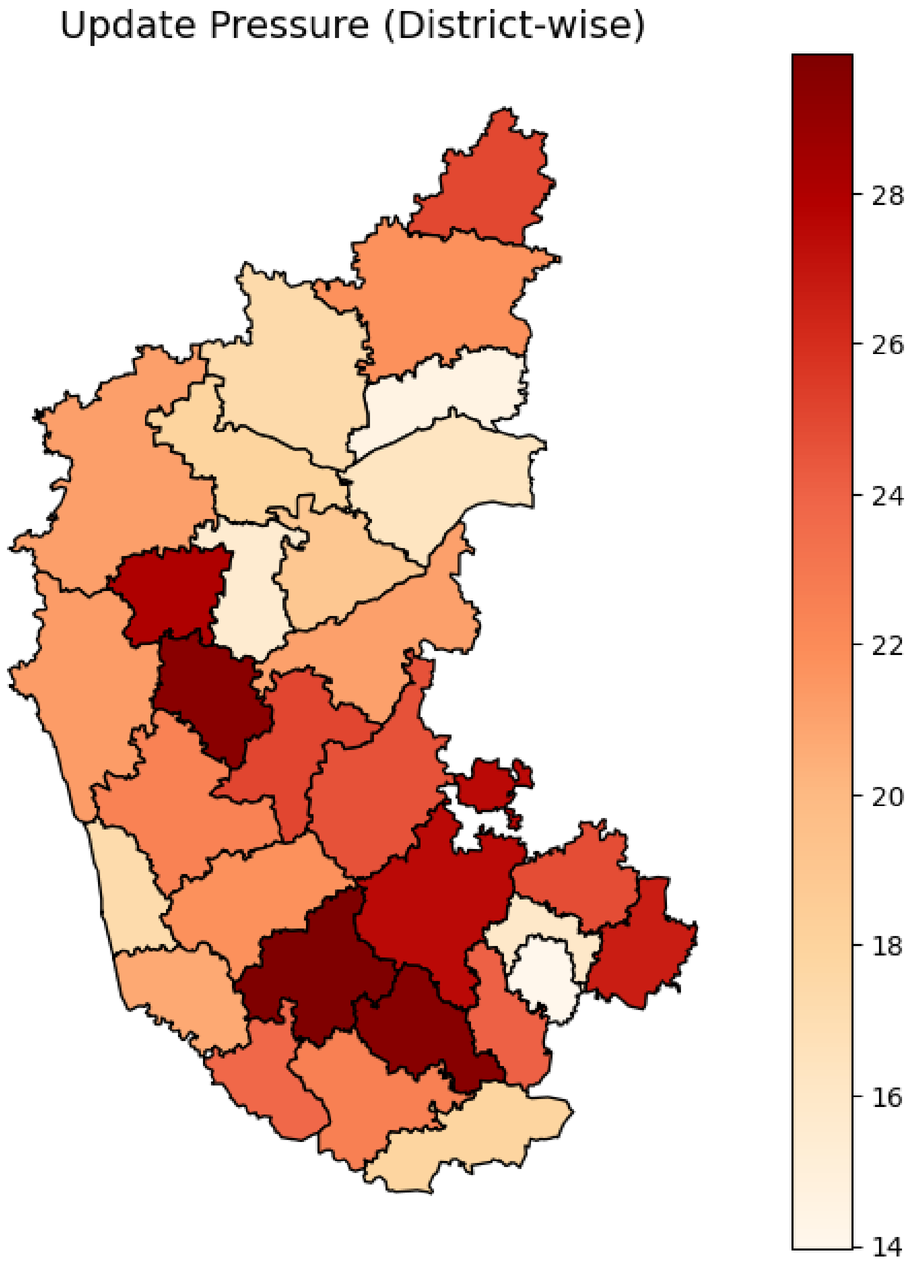
```
BASE_PATH = Path(".")

# load datasets
enrolment_df = load_activity_df("api_data_aadhar_enrolment", "enrolment")
demographic_df = load_activity_df("api_data_aadhar_demographic", "demographic_update")
biometric_df = load_activity_df("api_data_aadhar_biometric", "biometric_update")

# Karnataka only filter
ka_enrolment_df = enrolment_df.loc[enrolment_df["state"] == "Karnataka"].copy()
ka_demographic_df = demographic_df.loc[demographic_df["state"] == "Karnataka"].copy()
ka_biometric_df = biometric_df.loc[biometric_df["state"] == "Karnataka"].copy()

# geojson read for KA
ka_geo = gpd.read_file("mapKarnataka.json")

# district name normalization (removing extra spaces, full lwoercase, unicode
normalization, etc)
for df in [ka_enrolment_df, ka_demographic_df, ka_biometric_df]:
    df.loc[:, "district_norm"] = df["district"].apply(normalize_text)

ka_geo["district_norm"] = ka_geo["district"].apply(normalize_text)
canonical_districts = ka_geo["district_norm"].unique().tolist()

# all unique district names from our data (Aadhaar)
all_districts = (
    pd.concat([
        ka_enrolment_df["district_norm"],
        ka_demographic_df["district_norm"],
        ka_biometric_df["district_norm"]
    ])
    .dropna()
    .unique()
)

# fuzzymatch district names to canonical names in the geoJSON found
district_lookup = build_district_lookup(all_districts, canonical_districts, threshold=85)

# manual corrections for mismatches
# Karnataka has a lot of renamed districts (Bangalore→Bengaluru, Belgaum→Belagavi, etc.)
# also the geoJSON found didn't have Vijayanagara in it! (it was made 6 years ago)
manual_map = {
    "bangalore": "bengaluru urban",
    "bangalore rural": "bengaluru rural",
    "bengaluru": "bengaluru urban",
    "bengaluru south": "bengaluru urban",
    "belgaum": "belagavi",
    "bellary": "ballari",
    "bijapur": "vijayapura",
    "gulbarga": "kalaburagi",
    "mysore": "mysuru",
    "shimoga": "shivamogga",
    "chickmagalur": "chikmagalur",
    "bijapur kar": "vijayapura",
    "vijayanagara": "ballari",  # new district from Ballari, 6 year old geojson
}

district_lookup.update(manual_map)

# mapping district names
for df in [ka_enrolment_df, ka_demographic_df, ka_biometric_df]:
    df.loc[:, "district_matched"] = df["district_norm"].map(district_lookup)

# calculate totals by summing age groups
ka_enrolment_df.loc[:, "enrolment_total"] = (
    ka_enrolment_df["age_0_5"]
    + ka_enrolment_df["age_5_17"]
    + ka_enrolment_df["age_18_greater"]
)

ka_demographic_df.loc[:, "demographic_update_total"] = (
    ka_demographic_df["demo_age_5_17"]
    + ka_demographic_df["demo_age_17_"]
)

ka_biometric_df.loc[:, "biometric_update_total"] = (
    ka_biometric_df["bio_age_5_17"]
    + ka_biometric_df["bio_age_17_"]
)

# district-wise aggregation
district_enrolment = ka_enrolment_df.groupby("district_matched",
as_index=False)["enrolment_total"].sum()
district_demo = ka_demographic_df.groupby("district_matched",
as_index=False)["demographic_update_total"].sum()
district_bio = ka_biometric_df.groupby("district_matched",
as_index=False)["biometric_update_total"].sum()

# merge everything together
district_wide = (
    district_enrolment
    .merge(district_demo, on="district_matched", how="left")
    .merge(district_bio, on="district_matched", how="left")
    .fillna(0)
)

# calculate derived metrics
district_wide["total_updates"] = (
    district_wide["demographic_update_total"] + district_wide["biometric_update_total"]
)

district_wide["total_activity"] = (
    district_wide["enrolment_total"] + district_wide["total_updates"]
)

# UPDATE PRESSURE: updates per enrolment
```

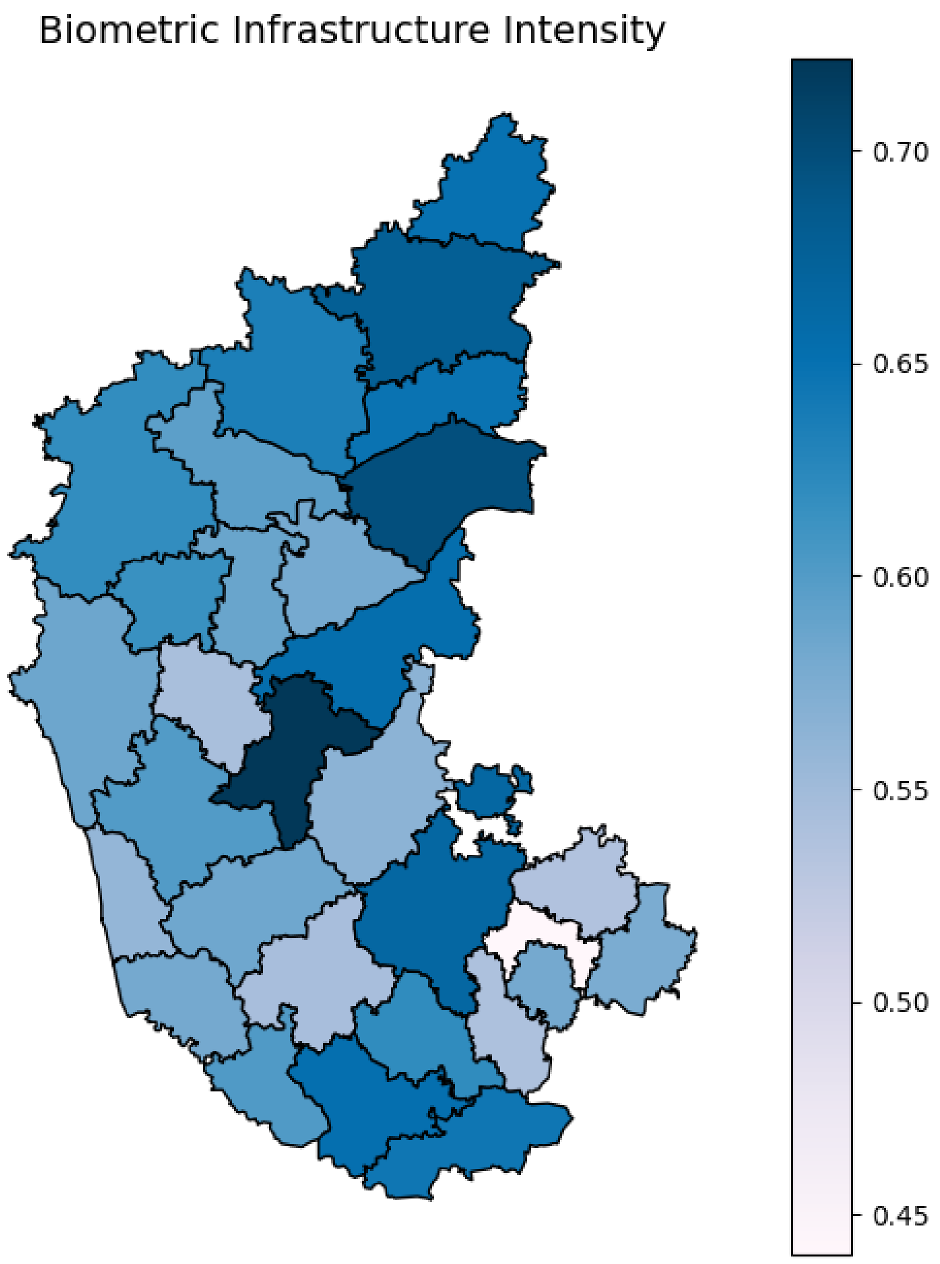
```
district_wide["update_pressure"] = (
    district_wide["total_updates"] / district_wide["enrolment_total"]
)

# What fraction of updates are biometric? (indicates infra needs)
district_wide["biometric_infra_intensity"] = (
    district_wide["biometric_update_total"] / district_wide["total_updates"]
)

# Quick look at top districts
print("Top 5 Districts by Total Aadhaar Activity:")
top5_activity = (
    district_wide
    .sort_values("total_activity", ascending=False)
    .loc[:, [
        "district_matched",
        "enrolment_total",
        "total_updates",
        "total_activity"
    ]]
    .head(5)
    .reset_index(drop=True)
)

top5_activity
```

```
Top 5 Districts by Total Aadhaar Activity:

  district_matched  enrolment_total  total_updates  total_activity
0  bengaluru urban            61273         855680          916953
1         belagavi            14548         307413          321961
2       kalaburagi             9138         198521          207659
3          ballari             9120         192026          201146
4       vijayapura             9830         170569          180399
```

```
# merge metrics with geodata
ka_geo_metrics = ka_geo.merge(
    district_wide,
    left_on="district_norm",
    right_on="district_matched",
    how="left"
)

# visualizations
plot_choropleth(ka_geo_metrics, "update_pressure", "Update Pressure (District-wise)",
"OrRd")
plot_choropleth(ka_geo_metrics, "biometric_infra_intensity", "Biometric Infrastructure
Intensity", "PuBu")
```

```
<Axes: title={'center': 'Biometric Infrastructure Intensity'}>
```


Update Pressure (District-wise)
28
26
24
22
20
18
16
14



```python
# Outlier detection using IQR method (more robust than z-score for smaller samples)
q1 = district_wide["update_pressure"].quantile(0.25)
q3 = district_wide["update_pressure"].quantile(0.75)
iqr = q3 - q1

district_wide["high_pressure_outlier"] = (
    district_wide["update_pressure"] > (q3 + 1.5 * iqr)
)

outliers = district_wide[district_wide["high_pressure_outlier"]]
print(f"High-pressure outlier districts: {len(outliers)}")
if len(outliers) > 0:
    print(outliers[["district_matched", "update_pressure"]])
else:
    print("No outliers found - Karnataka districts have relatively uniform update patterns!")
```

```
High-pressure outlier districts: 0
No outliers found - Karnataka districts have relatively uniform update patterns!
```