V4_GDI

October 16, 2025

1 Gentrification Degree Index (GDI) for Paris (2013-2021)

1.1 Version 4.0 - Comprehensive Methodology Implementation

Research Context: Analyzing gentrification patterns in Paris intra-muros using a theoretically-grounded composite index.

Methodology: Following the academic framework for measuring gentrification through multi-dimensional socio-economic indicators at IRIS level.

Time Period: 2013, 2017, 2021 (three observation points)

Spatial Unit: IRIS (Îlots Regroupés pour l'Information Statistique) - French infra-urban statistical units of ~2,000 inhabitants

---

1.1.1 Theoretical Background

Gentrification is defined as the socio-spatial transformation of historically working-class urban areas through: - **Income uplift**: Rising median disposable income - **Class recomposition**: Influx of professionals/executives, displacement of manual workers - **Demographic shifts**: Young professionals (25-39) replacing elderly populations - **Economic profile change**: Shift from welfare/pension to labor income sources

This notebook implements the GDI formula:

$$GDI_{i,t} = \frac{1}{N}(Z^{(t)}_{medinc} + Z^{(t)}_{CS3} - Z^{(t)}_{CS6} + Z^{(t)}_{25-39} - Z^{(t)}_{65+} + Z^{(t)}_{labor} - Z^{(t)}_{pens} - Z^{(t)}_{social})$$

Where: - Z = year-specific standardized (z-score) values - N = 8 components - Positive terms indicate gentrification - Negative terms indicate absence of gentrification

1.2 1. Setup and Configuration

[11]:
```python
# Import libraries
import pandas as pd
import numpy as np
import geopandas as gpd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
from pathlib import Path
from scipy import stats
import warnings
warnings.filterwarnings('ignore')

# Configuration
plt.style.use('seaborn-v0_8-darkgrid')
sns.set_palette("husl")
%matplotlib inline

# Paths
DATA_DIR = Path('outputs/clean_v3')
FIGURES_DIR = Path('outputs/figures_gdi')
TABLES_DIR = Path('outputs/tables_gdi')
MAPS_DIR = Path('outputs/maps_gdi')

# Create directories
for dir_path in [FIGURES_DIR, TABLES_DIR, MAPS_DIR]:
    dir_path.mkdir(parents=True, exist_ok=True)

# Random seed for reproducibility
np.random.seed(42)

print("  Environment configured")
print(f"  Data directory: {DATA_DIR}")
print(f"  Output directories created: {FIGURES_DIR}, {TABLES_DIR}, {MAPS_DIR}")
```

```
 Environment configured
  Data directory: outputs/clean_v3
  Output directories created: outputs/figures_gdi, outputs/tables_gdi,
outputs/maps_gdi
```

1.3 2. Data Loading

Load all required datasets for the three observation years: 2013, 2017, 2021

[18]:
```
print("Loading datasets...\n")

# FILOSOFI (income data) - 3 years
filosofi_2013 = pd.read_parquet(Path('../datasets') / 'filosofi_2013_paris.
 ↪parquet')
filosofi_2017 = pd.read_parquet(Path('../datasets') / 'filosofi_2017_paris.
 ↪parquet')
filosofi_2021 = pd.read_parquet(Path('../datasets') / 'filosofi_2021_paris.
 ↪parquet')

print(f"FILOSOFI 2013: {filosofi_2013.shape}")
print(f"FILOSOFI 2017: {filosofi_2017.shape}")
```

```python
print(f"FILOSOFI 2021: {filosofi_2021.shape}")

# CENSUS (demographic/social data) - 3 years
census_2013 = pd.read_parquet(Path('../datasets') / 'census_2013_paris.parquet')
census_2017 = pd.read_parquet(Path('../datasets') / 'census_2017_paris.parquet')
census_2021 = pd.read_parquet(Path('../datasets') / 'census_2021_paris.parquet')

print(f"\nCENSUS 2013: {census_2013.shape}")
print(f"CENSUS 2017: {census_2017.shape}")
print(f"CENSUS 2021: {census_2021.shape}")

# IRIS geographic boundaries (all 992 IRIS)
iris_geo = gpd.read_file(Path('../outputs') / 'iris_paris75.geojson')
print(f"\nIRIS Geography: {iris_geo.shape}")
print(f"  CRS: {iris_geo.crs}")
print(f"  Columns: {list(iris_geo.columns)}")
```

```
Loading datasets…

FILOSOFI 2013: (853, 10)
FILOSOFI 2017: (871, 10)
FILOSOFI 2021: (992, 10)

CENSUS 2013: (992, 13)
CENSUS 2017: (992, 13)
CENSUS 2021: (992, 13)

IRIS Geography: (992, 10)
  CRS: EPSG:4326
  Columns: ['dep', 'insee_com', 'nom_com', 'iris', 'code_iris', 'nom_iris',
'typ_iris', 'geo_point_2d', 'id', 'geometry']
```

1.4 3. Data Preparation and Harmonization

1.4.1 3.1 Handle Missing Values (INSEE Suppression Codes)

Following INSEE methodology: - `'ns'` = non significatif (small population <1000, <500 households) - `'nd'` = non disponible (non-residential zones: parks, industrial) - `'s'`, `'c'` = secret statistique

[13]:
```python
def clean_filosofi_data(df, year):
    """
    Clean FILOSOFI data by converting suppression codes to NaN.
    Preserves only residential IRIS for gentrification analysis.
    """
    df_clean = df.copy()

    # Suppression codes to convert to NaN
    suppression_codes = ['ns', 'nd', 's', 'c', '/', '-', 'NA']

    # Identify numeric columns
    numeric_cols = ['median_uc', 'q1_uc', 'q3_uc', 'd9d1_ratio', 'gini',
                    'share_activity_income', 'share_pensions',␣
↪'share_social_benefits']

    for col in numeric_cols:
        if col in df_clean.columns:
            # Convert suppression codes to NaN
            mask = df_clean[col].isin(suppression_codes)
            df_clean.loc[mask, col] = np.nan

            # Convert to numeric
            df_clean[col] = df_clean[col].astype(str).str.replace(',', '.')
            df_clean[col] = pd.to_numeric(df_clean[col], errors='coerce')

    print(f"FILOSOFI {year}: {len(df_clean)} IRIS, {df_clean[numeric_cols].
↪isna().sum().sum()} total missing values")

    return df_clean

# Clean all FILOSOFI datasets
filosofi_2013_clean = clean_filosofi_data(filosofi_2013, 2013)
filosofi_2017_clean = clean_filosofi_data(filosofi_2017, 2017)
filosofi_2021_clean = clean_filosofi_data(filosofi_2021, 2021)
```

```
FILOSOFI 2013: 853 IRIS, 0 total missing values
FILOSOFI 2017: 871 IRIS, 8 total missing values
FILOSOFI 2021: 992 IRIS, 1024 total missing values
```

1.4.2 3.2 Calculate Census-Derived Variables

Transform census counts into percentages for GDI components: - **CS3 Share**: % of population 15+ who are executives/professionals (cadres) - **CS6 Share**: % of population 15+ who are manual workers (ouvriers) - **Age 25-39 Share**: % of total population - **Age 65+ Share**: % of total population

[14]:
```
def calculate_census_shares(df, year):
    """
    Calculate percentage shares from census counts.
    Handles different column names and avoids KeyError when expected
    share columns are missing. Uses 'pop_total' as total population
    (not 'population') and safeguards divisions by zero.
    """
    df_shares = df.copy()

    # Socio-professional shares (% of active population 15+)
    if 'pop_15plus' in df.columns and df['pop_15plus'].sum() > 0:
        denom_15 = df['pop_15plus'].replace({0: np.nan})
        df_shares['share_cs3'] = (df.get('pop_cadres', 0) / denom_15 * 100)
        df_shares['share_cs6'] = (df.get('pop_ouvriers', 0) / denom_15 * 100)
    else:
        df_shares['share_cs3'] = np.nan
        df_shares['share_cs6'] = np.nan

    # Age shares (% of total population) - use 'pop_total' column in this␣
 ↪dataset
    if 'pop_total' in df.columns and df['pop_total'].sum() > 0:
        denom_tot = df['pop_total'].replace({0: np.nan})
        df_shares['share_25_39'] = (df.get('pop_25_39', 0) / denom_tot * 100)
        df_shares['share_65plus'] = (df.get('pop_65plus', 0) / denom_tot * 100)
    else:
        df_shares['share_25_39'] = np.nan
        df_shares['share_65plus'] = np.nan

    # Helper to safely format means
    def safe_mean(col):
        if col in df_shares.columns and df_shares[col].notna().any():
            return f"{df_shares[col].mean():.1f}%"
        return "N/A"

    print(f"CENSUS {year}: Calculated shares for {len(df_shares)} IRIS")
    print(f"  Mean CS3 (executives): {safe_mean('share_cs3')}")
    print(f"  Mean CS6 (workers):   {safe_mean('share_cs6')}")
    print(f"  Mean Age 25-39:       {safe_mean('share_25_39')}")
    print(f"  Mean Age 65+:         {safe_mean('share_65plus')}")

    return df_shares

# Calculate shares for all years
census_2013_shares = calculate_census_shares(census_2013, 2013)
census_2017_shares = calculate_census_shares(census_2017, 2017)
census_2021_shares = calculate_census_shares(census_2021, 2021)
```

```
CENSUS 2013: Calculated shares for 992 IRIS
  Mean CS3 (executives): 28.3%
  Mean CS6 (workers):   4.7%
  Mean Age 25-39:       26.1%
  Mean Age 65+:         15.3%
CENSUS 2017: Calculated shares for 992 IRIS
  Mean CS3 (executives): 29.1%
  Mean CS6 (workers):   4.2%
  Mean Age 25-39:       25.6%
  Mean Age 65+:         16.7%
CENSUS 2021: Calculated shares for 992 IRIS
```

```
Mean CS3 (executives): 30.7%
Mean CS6 (workers):    4.0%
Mean Age 25-39:        25.6%
Mean Age 65+:          17.2%
```

1.4.3 3.3 Merge Datasets by Year

Combine FILOSOFI + CENSUS + Geography for each observation year

[19]:
```
def merge_year_data(filosofi_df, census_df, iris_geo_df, year):
    """
    Merge all data sources for a given year on code_iris.
    """
    # Start with IRIS geography (ensures all IRIS are included)
    merged = iris_geo_df[['code_iris', 'nom_iris', 'typ_iris', 'geometry']].
↪copy()
    merged = merged.rename(columns={'nom_iris': 'libelle_iris'})

    # Merge FILOSOFI
    merged = merged.merge(
        filosofi_df[['code_iris', 'median_uc', 'q1_uc', 'q3_uc', 'd9d1_ratio',␣
↪'gini',
                     'share_activity_income', 'share_pensions',␣
↪'share_social_benefits']],
        on='code_iris',
        how='left'
    )

    # Merge CENSUS shares
    merged = merged.merge(
        census_df[['code_iris', 'share_cs3', 'share_cs6', 'share_25_39',␣
↪'share_65plus']],
        on='code_iris',
        how='left'
    )

    # Add year column
    merged['year'] = year

    # Filter to residential IRIS only (typ_iris == 'H')
    merged_residential = merged[merged['typ_iris'] == 'H'].copy()

    print(f"Year {year}:")
    print(f"  Total IRIS: {len(merged)}")
    print(f"  Residential IRIS (H): {len(merged_residential)}")
    print(f"  Complete cases (no missing): {merged_residential.dropna().
↪shape[0]}")
```

```
    return merged_residential

# Merge data for all three years
data_2013 = merge_year_data(filosofi_2013_clean, census_2013_shares, iris_geo,␣
 ↪2013)
data_2017 = merge_year_data(filosofi_2017_clean, census_2017_shares, iris_geo,␣
 ↪2017)
data_2021 = merge_year_data(filosofi_2021_clean, census_2021_shares, iris_geo,␣
 ↪2021)
```

```
Year 2013:
  Total IRIS: 992
  Residential IRIS (H): 861
  Complete cases (no missing): 853
Year 2017:
  Total IRIS: 992
  Residential IRIS (H): 861
  Complete cases (no missing): 855
Year 2021:
  Total IRIS: 992
  Residential IRIS (H): 861
  Complete cases (no missing): 852
```

1.5 4. GDI Component Standardization

1.5.1 Year-Specific Z-Score Normalization

Critical: Each year is standardized independently to capture **relative position within Paris** for that year.

This allows comparison of: - An IRIS's standing relative to citywide distribution in each year - Changes in relative position over time (gentrification trajectory)

Formula: $Z = \frac{X-\mu}{\sigma}$

[20]:
```
def standardize_gdi_components(df, year):
    """
    Calculate z-scores for all 8 GDI components within the year's distribution.
    Returns dataframe with original values + z-scores.
    """
    df_std = df.copy()

    # Define the 8 GDI components
    components = {
        'z_median_income': 'median_uc',
        'z_cs3': 'share_cs3',
        'z_cs6': 'share_cs6',
        'z_age_25_39': 'share_25_39',
        'z_age_65plus': 'share_65plus',
        'z_labor_income': 'share_activity_income',
        'z_pension_income': 'share_pensions',
        'z_social_benefits': 'share_social_benefits'
    }

    print(f"\nStandardizing {year} data:")
    print("-" * 60)

    for z_col, raw_col in components.items():
        if raw_col in df.columns:
            # Calculate z-score (handling NaN)
            values = df[raw_col].dropna()
            mean = values.mean()
            std = values.std()

            df_std[z_col] = (df[raw_col] - mean) / std

            print(f"{raw_col:30s} → {z_col:20s} ( ={mean:.2f},  ={std:.2f})")

    return df_std

# Standardize all years
data_2013_std = standardize_gdi_components(data_2013, 2013)
data_2017_std = standardize_gdi_components(data_2017, 2017)
data_2021_std = standardize_gdi_components(data_2021, 2021)
```

```
Standardizing 2013 data:
------------------------------------------------------------
median_uc                      → z_median_income      ( =28587.41,  =8693.04)
share_cs3                      → z_cs3                ( =28.38,  =8.71)
share_cs6                      → z_cs6                ( =4.66,  =2.92)
share_25_39                    → z_age_25_39          ( =26.11,  =6.84)
share_65plus                   → z_age_65plus         ( =15.58,  =5.18)
share_activity_income          → z_labor_income       ( =76.97,  =8.25)
share_pensions                 → z_pension_income     ( =19.83,  =5.36)
share_social_benefits          → z_social_benefits    ( =3.56,  =3.01)

Standardizing 2017 data:
------------------------------------------------------------
median_uc                      → z_median_income      ( =29768.58,  =8566.50)
share_cs3                      → z_cs3                ( =29.52,  =8.82)
share_cs6                      → z_cs6                ( =4.22,  =2.53)
share_25_39                    → z_age_25_39          ( =25.82,  =6.93)
share_65plus                   → z_age_65plus         ( =16.97,  =5.21)
share_activity_income          → z_labor_income       ( =87.46,  =8.59)
share_pensions                 → z_pension_income     ( =20.14,  =5.50)
```

```
share_social_benefits          → z_social_benefits    ( =3.59,  =3.27)

Standardizing 2021 data:
------------------------------------------------------------
median_uc                      → z_median_income      ( =32197.43,  =8956.91)
share_cs3                      → z_cs3                ( =30.86,  =9.17)
share_cs6                      → z_cs6                ( =3.94,  =2.51)
share_25_39                    → z_age_25_39          ( =25.74,  =7.12)
share_65plus                   → z_age_65plus         ( =17.58,  =5.12)
share_activity_income          → z_labor_income       ( =87.85,  =10.23)
share_pensions                 → z_pension_income     ( =19.51,  =5.43)
share_social_benefits          → z_social_benefits    ( =3.45,  =3.17)
```

1.6 5. Calculate Gentrification Degree Index (GDI)

1.6.1 GDI Formula Implementation

$$GDI = \frac{1}{8}(Z_{income} + Z_{CS3} - Z_{CS6} + Z_{25-39} - Z_{65+} + Z_{labor} - Z_{pension} - Z_{social})$$

Interpretation: - **GDI $>$ 0**: Above-average gentrification (more affluent, professional, young) - **GDI $=$ 0**: Average neighborhood profile - **GDI $<$ 0**: Below-average gentrification (working-class, elderly, welfare-dependent) - **GDI $>$ +1.0**: Highly gentrified (top ~15%) - **GDI $<$ -1.0**: Least gentrified (bottom ~15%)

```
[21]: def calculate_gdi(df, year):
          """
          Calculate GDI as the mean of 8 standardized components (with appropriate␣
       ↪signs).
          """
          df_gdi = df.copy()

          # Calculate GDI with equal weights (1/8 each)
          df_gdi['GDI'] = (
              df['z_median_income'] +       # Positive: higher income = more gentrified
              df['z_cs3'] -                 # Positive: more executives = more␣
       ↪gentrified
              df['z_cs6'] +                 # Negative: fewer workers = more gentrified
              df['z_age_25_39'] -           # Positive: more young adults = more␣
       ↪gentrified
              df['z_age_65plus'] +          # Negative: fewer elderly = more gentrified
              df['z_labor_income'] -        # Positive: more work income = more␣
       ↪gentrified
              df['z_pension_income'] -      # Negative: less pension = more gentrified
              df['z_social_benefits']       # Negative: less welfare = more gentrified
          ) / 8

          # Calculate summary statistics
          gdi_stats = df_gdi['GDI'].describe()
```

```
    print(f"\n{'='*60}")
    print(f"GDI CALCULATED FOR {year}")
    print(f"{'='*60}")
    print(f"\nDistribution:")
    print(f"  Mean:   {gdi_stats['mean']:6.3f}")
    print(f"  Std:    {gdi_stats['std']:6.3f}")
    print(f"  Min:    {gdi_stats['min']:6.3f}")
    print(f"  Q1:     {gdi_stats['25%']:6.3f}")
    print(f"  Median: {gdi_stats['50%']:6.3f}")
    print(f"  Q3:     {gdi_stats['75%']:6.3f}")
    print(f"  Max:    {gdi_stats['max']:6.3f}")

    # Count extreme cases
    high_gentri = (df_gdi['GDI'] > 1.0).sum()
    low_gentri = (df_gdi['GDI'] < -1.0).sum()

    print(f"\nExtreme Cases:")
    print(f"  High gentrification (GDI > 1.0):  {high_gentri} IRIS␣
  ↪({high_gentri/len(df_gdi)*100:.1f}%)")
    print(f"  Low gentrification (GDI < -1.0):  {low_gentri} IRIS ({low_gentri/
  ↪len(df_gdi)*100:.1f}%)")

    return df_gdi

# Calculate GDI for all years
data_2013_gdi = calculate_gdi(data_2013_std, 2013)
data_2017_gdi = calculate_gdi(data_2017_std, 2017)
data_2021_gdi = calculate_gdi(data_2021_std, 2021)
```

```
============================================================
GDI CALCULATED FOR 2013
============================================================

Distribution:
  Mean:   -0.001
  Std:     0.600
  Min:    -2.104
  Q1:     -0.295
  Median:  0.083
  Q3:      0.422
  Max:     1.221

Extreme Cases:
  High gentrification (GDI > 1.0):  12 IRIS (1.4%)
  Low gentrification (GDI < -1.0):  57 IRIS (6.6%)
```

```
============================================================
GDI CALCULATED FOR 2017
============================================================

Distribution:
  Mean:   -0.000
  Std:     0.662
  Min:    -2.366
  Q1:     -0.321
  Median:  0.109
  Q3:      0.451
  Max:     1.307

Extreme Cases:
  High gentrification (GDI > 1.0):  23 IRIS (2.7%)
  Low gentrification (GDI < -1.0):  74 IRIS (8.6%)

============================================================
GDI CALCULATED FOR 2021
============================================================

Distribution:
  Mean:    0.001
  Std:     0.653
  Min:    -2.226
  Q1:     -0.298
  Median:  0.082
  Q3:      0.449
  Max:     1.308

Extreme Cases:
  High gentrification (GDI > 1.0):  22 IRIS (2.6%)
  Low gentrification (GDI < -1.0):  71 IRIS (8.2%)
```

1.7 6. GDI Classification into Four Classes

1.7.1 Quartile-Based Classification

Each year's GDI distribution is divided into 4 classes using quartiles:

1. **Low Gentrification** (Q1, bottom 25%): Working-class, welfare-dependent, elderly populations
2. **Lower-Intermediate** (Q1-Q2): Below median, modest neighborhoods
3. **Upper-Intermediate** (Q2-Q3): Above median, actively gentrifying areas
4. **High Gentrification** (Q4, top 25%): Affluent, professional, young populations

[22]:
```python
def classify_gdi_quartiles(df, year):
    """
    Classify IRIS into 4 gentrification classes based on GDI quartiles.
    """
    df_class = df.copy()

    # Calculate quartile breakpoints
    q1 = df['GDI'].quantile(0.25)
    q2 = df['GDI'].quantile(0.50)  # median
    q3 = df['GDI'].quantile(0.75)

    # Classify into 4 classes
    conditions = [
        df['GDI'] <= q1,
        (df['GDI'] > q1) & (df['GDI'] <= q2),
        (df['GDI'] > q2) & (df['GDI'] <= q3),
        df['GDI'] > q3
    ]

    labels = ['Low', 'Lower-Intermediate', 'Upper-Intermediate', 'High']

    df_class['GDI_class'] = np.select(conditions, labels, default='Unknown')

    # Convert to categorical with proper ordering
    df_class['GDI_class'] = pd.Categorical(
        df_class['GDI_class'],
        categories=['Low', 'Lower-Intermediate', 'Upper-Intermediate', 'High'],
        ordered=True
    )

    print(f"\n{'='*60}")
    print(f"GDI CLASSIFICATION FOR {year}")
    print(f"{'='*60}")
    print(f"\nQuartile Breakpoints:")
    print(f"  Q1 (25th percentile): {q1:.3f}")
    print(f"  Q2 (median):          {q2:.3f}")
    print(f"  Q3 (75th percentile): {q3:.3f}")

    print(f"\nClass Distribution:")
    class_counts = df_class['GDI_class'].value_counts().sort_index()
    for cls, count in class_counts.items():
        pct = count / len(df_class) * 100
        print(f"  {cls:25s}: {count:4d} IRIS ({pct:5.1f}%)")

    return df_class

# Classify all years
```

```
data_2013_class = classify_gdi_quartiles(data_2013_gdi, 2013)
data_2017_class = classify_gdi_quartiles(data_2017_gdi, 2017)
data_2021_class = classify_gdi_quartiles(data_2021_gdi, 2021)
```

```
============================================================
GDI CLASSIFICATION FOR 2013
============================================================

Quartile Breakpoints:
  Q1 (25th percentile): -0.295
  Q2 (median):          0.083
  Q3 (75th percentile): 0.422

Class Distribution:
  Low                      :  214 IRIS ( 24.9%)
  Lower-Intermediate       :  213 IRIS ( 24.7%)
  Upper-Intermediate       :  213 IRIS ( 24.7%)
  High                     :  213 IRIS ( 24.7%)

============================================================
GDI CLASSIFICATION FOR 2017
============================================================

Quartile Breakpoints:
  Q1 (25th percentile): -0.321
  Q2 (median):          0.109
  Q3 (75th percentile): 0.451

Class Distribution:
  Low                      :  214 IRIS ( 24.9%)
  Lower-Intermediate       :  214 IRIS ( 24.9%)
  Upper-Intermediate       :  213 IRIS ( 24.7%)
  High                     :  214 IRIS ( 24.9%)

============================================================
GDI CLASSIFICATION FOR 2021
============================================================

Quartile Breakpoints:
  Q1 (25th percentile): -0.298
  Q2 (median):          0.082
  Q3 (75th percentile): 0.449

Class Distribution:
  Low                      :  213 IRIS ( 24.7%)
  Lower-Intermediate       :  213 IRIS ( 24.7%)
  Upper-Intermediate       :  213 IRIS ( 24.7%)
```

```
High                        :  213 IRIS ( 24.7%)
```

1.8 7. Temporal Change Analysis (2013-2021)

1.8.1 Gentrification Trajectory Classification

Identify neighborhoods by their evolution pattern:

- **Intensifying**: Consistent significant upward GDI trend ($\Delta > 0$, $\Delta > 0$, $\Delta > 0.5$)
- **Declining**: Consistent significant downward GDI trend ($\Delta < 0$, $\Delta < 0$, $\Delta <$ -0.5)
- **Stable**: No significant change or inconsistent pattern ($|\Delta|$ 0.5)

Where: - Δ = GDI(2017) - GDI(2013) - Δ = GDI(2021) - GDI(2017) - Δ = GDI(2021) - GDI(2013)

[23]:
```
# Merge all years for temporal analysis
temporal_data = data_2013_class[['code_iris', 'libelle_iris', 'GDI',␣
 ↪'GDI_class', 'geometry']].rename(
    columns={'GDI': 'GDI_2013', 'GDI_class': 'class_2013'}
)

temporal_data = temporal_data.merge(
    data_2017_class[['code_iris', 'GDI', 'GDI_class']].rename(
        columns={'GDI': 'GDI_2017', 'GDI_class': 'class_2017'}
    ),
    on='code_iris',
    how='inner'
)

temporal_data = temporal_data.merge(
    data_2021_class[['code_iris', 'GDI', 'GDI_class']].rename(
        columns={'GDI': 'GDI_2021', 'GDI_class': 'class_2021'}
    ),
    on='code_iris',
    how='inner'
)

# Calculate changes
temporal_data['delta_1'] = temporal_data['GDI_2017'] - temporal_data['GDI_2013']
temporal_data['delta_2'] = temporal_data['GDI_2021'] - temporal_data['GDI_2017']
temporal_data['delta_total'] = temporal_data['GDI_2021'] -␣
 ↪temporal_data['GDI_2013']

# Threshold: 0.5 standard deviations
threshold = 0.5

# Classify trajectories
conditions = [
    # Intensifying: both periods positive AND total > threshold
    (temporal_data['delta_1'] > 0) & (temporal_data['delta_2'] > 0) &␣
 ↪(temporal_data['delta_total'] > threshold),

    # Declining: both periods negative AND total < -threshold
    (temporal_data['delta_1'] < 0) & (temporal_data['delta_2'] < 0) &␣
 ↪(temporal_data['delta_total'] < -threshold),
]

labels = ['Intensifying', 'Declining']

temporal_data['trajectory'] = np.select(conditions, labels, default='Stable')

print("\n" + "="*60)
print("TEMPORAL CHANGE ANALYSIS (2013-2021)")
print("="*60)

print(f"\nTotal IRIS analyzed: {len(temporal_data)}")
print(f"Threshold for significant change: ±{threshold} ")

print(f"\nTrajectory Distribution:")
traj_counts = temporal_data['trajectory'].value_counts()
for traj, count in traj_counts.items():
    pct = count / len(temporal_data) * 100
    mean_delta = temporal_data[temporal_data['trajectory'] ==␣
 ↪traj]['delta_total'].mean()
    print(f"  {traj:15s}: {count:4d} IRIS ({pct:5.1f}%) - Mean Δ = {mean_delta:
 ↪+.3f}")

print(f"\nChange Statistics:")
print(f"  Mean GDI change (Δ):     {temporal_data['delta_total'].mean():+.3f}")
print(f"  Std GDI change:           {temporal_data['delta_total'].std():.3f}")
print(f"  Max increase:             {temporal_data['delta_total'].max():+.3f}")
print(f"  Max decrease:             {temporal_data['delta_total'].min():+.3f}")
```

```
============================================================
TEMPORAL CHANGE ANALYSIS (2013-2021)
============================================================

Total IRIS analyzed: 861
Threshold for significant change: ±0.5

Trajectory Distribution:
  Stable         :  827 IRIS ( 96.1%) - Mean Δ = +0.005
  Declining      :   20 IRIS (  2.3%) - Mean Δ = -0.656
  Intensifying   :   14 IRIS (  1.6%) - Mean Δ = +0.588
```

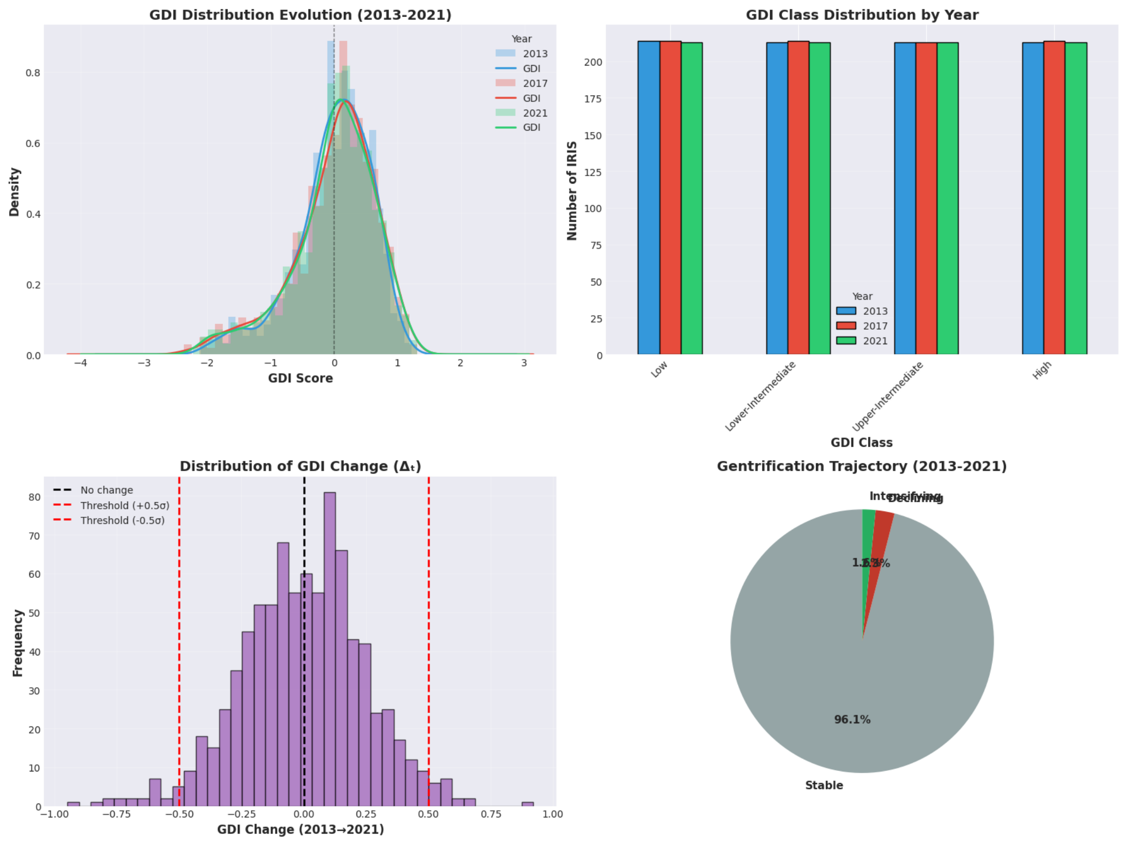
```
Change Statistics:
  Mean GDI change (Δ ):      -0.001
  Std GDI change:             0.248
  Max increase:              +0.921
  Max decrease:              -0.948
```

1.9 8. Visualizations

1.9.1 8.1 GDI Distribution Evolution

[24]:
```python
fig, axes = plt.subplots(2, 2, figsize=(16, 12))

# Plot 1: GDI distributions by year (KDE + histogram)
ax = axes[0, 0]
for year, data, color in [(2013, data_2013_class, '#3498db'),
                           (2017, data_2017_class, '#e74c3c'),
                           (2021, data_2021_class, '#2ecc71')]:
    data['GDI'].hist(bins=30, alpha=0.3, color=color, ax=ax, density=True,␣
 ↪label=str(year))
    data['GDI'].plot(kind='kde', ax=ax, color=color, linewidth=2)

ax.axvline(0, color='black', linestyle='--', linewidth=1, alpha=0.5)
ax.set_xlabel('GDI Score', fontsize=12, fontweight='bold')
ax.set_ylabel('Density', fontsize=12, fontweight='bold')
ax.set_title('GDI Distribution Evolution (2013-2021)', fontsize=14,␣
 ↪fontweight='bold')
ax.legend(title='Year', fontsize=10)
ax.grid(alpha=0.3)

# Plot 2: Class composition by year
ax = axes[0, 1]
class_evolution = pd.DataFrame({
    '2013': data_2013_class['GDI_class'].value_counts().sort_index(),
    '2017': data_2017_class['GDI_class'].value_counts().sort_index(),
    '2021': data_2021_class['GDI_class'].value_counts().sort_index()
})
class_evolution.plot(kind='bar', ax=ax, color=['#3498db', '#e74c3c',␣
 ↪'#2ecc71'], edgecolor='black')
ax.set_xlabel('GDI Class', fontsize=12, fontweight='bold')
ax.set_ylabel('Number of IRIS', fontsize=12, fontweight='bold')
ax.set_title('GDI Class Distribution by Year', fontsize=14, fontweight='bold')
ax.legend(title='Year', fontsize=10)
ax.set_xticklabels(ax.get_xticklabels(), rotation=45, ha='right')
ax.grid(axis='y', alpha=0.3)

# Plot 3: GDI change distribution
ax = axes[1, 0]
```

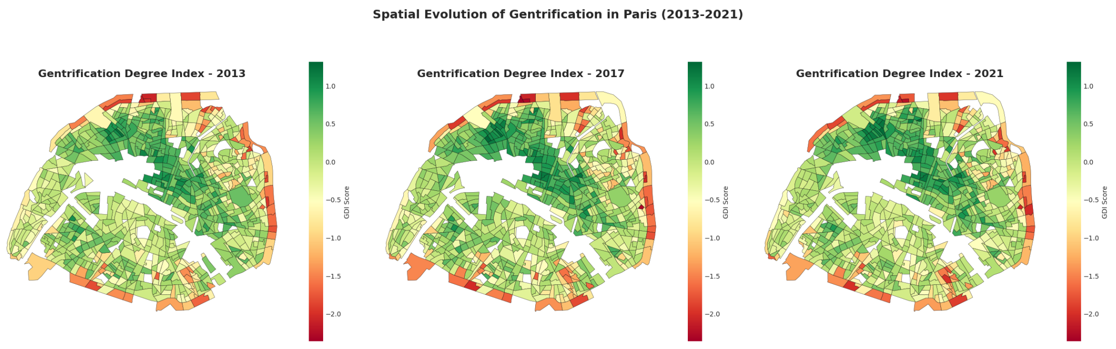
```python
temporal_data['delta_total'].hist(bins=40, ax=ax, color='#9b59b6',␣
 ↪edgecolor='black', alpha=0.7)
ax.axvline(0, color='black', linestyle='--', linewidth=2, label='No change')
ax.axvline(threshold, color='red', linestyle='--', linewidth=2,␣
 ↪label=f'Threshold (+{threshold} )')
ax.axvline(-threshold, color='red', linestyle='--', linewidth=2,␣
 ↪label=f'Threshold (-{threshold} )')
ax.set_xlabel('GDI Change (2013→2021)', fontsize=12, fontweight='bold')
ax.set_ylabel('Frequency', fontsize=12, fontweight='bold')
ax.set_title('Distribution of GDI Change (Δ )', fontsize=14, fontweight='bold')
ax.legend(fontsize=10)
ax.grid(alpha=0.3)

# Plot 4: Trajectory pie chart
ax = axes[1, 1]
traj_counts = temporal_data['trajectory'].value_counts()
colors = {'Intensifying': '#27ae60', 'Stable': '#95a5a6', 'Declining':␣
 ↪'#c0392b'}
wedges, texts, autotexts = ax.pie(
    traj_counts.values,
    labels=traj_counts.index,
    autopct='%1.1f%%',
    colors=[colors.get(t, '#bdc3c7') for t in traj_counts.index],
    startangle=90,
    textprops={'fontsize': 11, 'fontweight': 'bold'}
)
ax.set_title('Gentrification Trajectory (2013-2021)', fontsize=14,␣
 ↪fontweight='bold')

plt.tight_layout()
plt.savefig(FIGURES_DIR / 'gdi_evolution_overview.png', dpi=300,␣
 ↪bbox_inches='tight')
plt.show()

print(f"  Figure saved: {FIGURES_DIR / 'gdi_evolution_overview.png'}")
```



```
Figure saved: outputs/figures_gdi/gdi_evolution_overview.png
```

1.9.2 8.2 Spatial Maps of GDI

[25]:
```python
# Convert to GeoDataFrame for mapping
temporal_gdf = gpd.GeoDataFrame(temporal_data, geometry='geometry', crs='EPSG:
↪4326')

# Create 3-panel map: 2013, 2017, 2021
fig, axes = plt.subplots(1, 3, figsize=(24, 8))

vmin = temporal_gdf[['GDI_2013', 'GDI_2017', 'GDI_2021']].min().min()
vmax = temporal_gdf[['GDI_2013', 'GDI_2017', 'GDI_2021']].max().max()

for idx, (year, gdi_col, ax) in enumerate([
    (2013, 'GDI_2013', axes[0]),
    (2017, 'GDI_2017', axes[1]),
    (2021, 'GDI_2021', axes[2])
]):
    temporal_gdf.plot(
        column=gdi_col,
        ax=ax,
        cmap='RdYlGn',
        legend=True,
        vmin=vmin,
        vmax=vmax,
        edgecolor='black',
        linewidth=0.3,
        legend_kwds={'label': 'GDI Score', 'shrink': 0.8}
    )
    ax.set_title(f'Gentrification Degree Index - {year}', fontsize=16,␣
 ↪fontweight='bold')
    ax.axis('off')

plt.suptitle('Spatial Evolution of Gentrification in Paris (2013-2021)',
             fontsize=18, fontweight='bold', y=0.98)
plt.tight_layout()
plt.savefig(MAPS_DIR / 'gdi_spatial_evolution.png', dpi=300,␣
 ↪bbox_inches='tight')
plt.show()

print(f" Map saved: {MAPS_DIR / 'gdi_spatial_evolution.png'}")
```

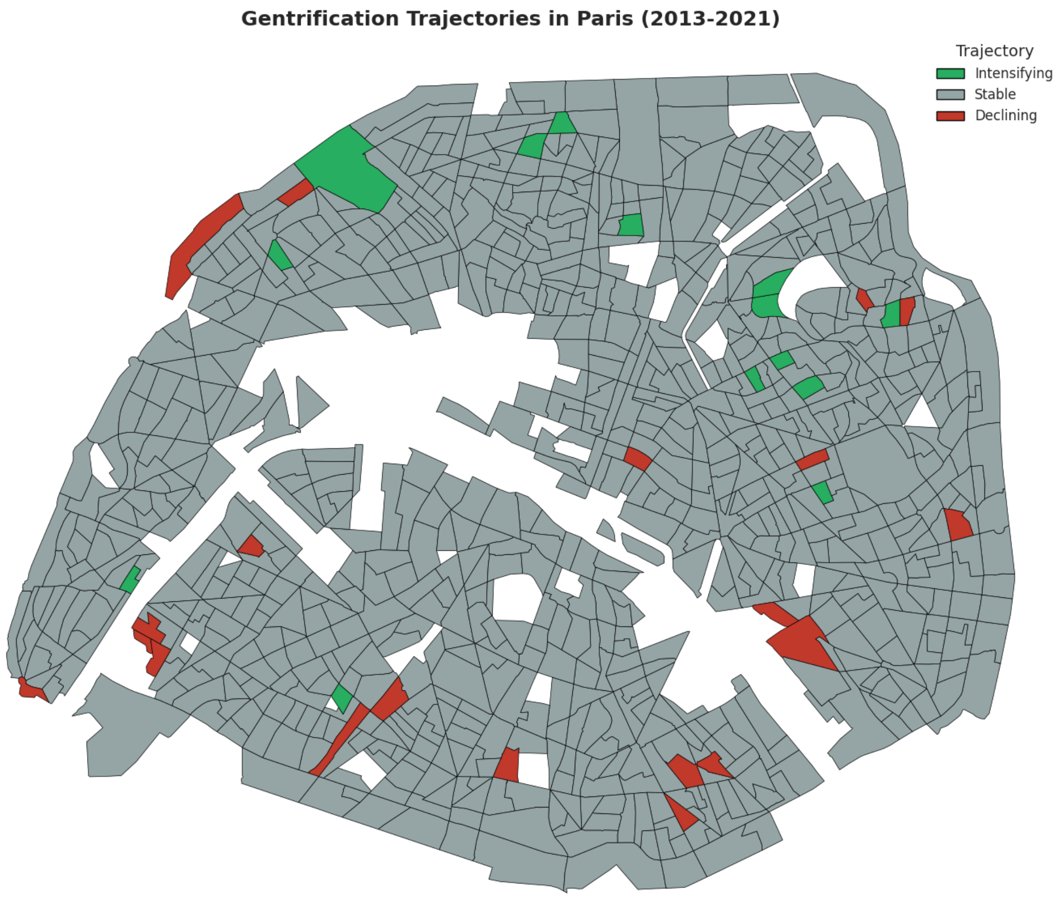

Spatial Evolution of Gentrification in Paris (2013-2021)
Gentrification Degree Index - 2013
Gentrification Degree Index - 2017
Gentrification Degree Index - 2021
GDI Score
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0

```
 Map saved: outputs/maps_gdi/gdi_spatial_evolution.png
```

1.9.3 8.3 Trajectory Map

[26]:
```
fig, ax = plt.subplots(1, 1, figsize=(14, 14))

# Color scheme for trajectories
color_map = {
    'Intensifying': '#27ae60',
    'Stable': '#95a5a6',
    'Declining': '#c0392b'
}
```

```
temporal_gdf['color'] = temporal_gdf['trajectory'].map(color_map)

temporal_gdf.plot(
    ax=ax,
    color=temporal_gdf['color'],
    edgecolor='black',
    linewidth=0.5
)

# Create legend manually
from matplotlib.patches import Patch
legend_elements = [Patch(facecolor=color, edgecolor='black', label=traj)
                   for traj, color in color_map.items()]
ax.legend(handles=legend_elements, loc='upper right', fontsize=12,␣
 ↪title='Trajectory', title_fontsize=14)

ax.set_title('Gentrification Trajectories in Paris (2013-2021)', fontsize=18,␣
 ↪fontweight='bold')
ax.axis('off')

plt.tight_layout()
plt.savefig(MAPS_DIR / 'gdi_trajectories_map.png', dpi=300, bbox_inches='tight')
plt.show()

print(f" Map saved: {MAPS_DIR / 'gdi_trajectories_map.png'}")
```



```
Map saved: outputs/maps_gdi/gdi_trajectories_map.png
```

1.10 9. Export Results

Save GDI scores, classifications, and trajectories for further analysis

[27]:
```python
# Export individual years
for year, data in [(2013, data_2013_class), (2017, data_2017_class), (2021,␣
↪data_2021_class)]:
    output_cols = ['code_iris', 'libelle_iris', 'typ_iris', 'GDI', 'GDI_class',
                   'median_uc', 'share_cs3', 'share_cs6', 'share_25_39',␣
↪'share_65plus',
                   'share_activity_income', 'share_pensions',␣
↪'share_social_benefits']

    data[output_cols].to_csv(TABLES_DIR / f'gdi_{year}.csv', index=False)
    print(f"  Exported: {TABLES_DIR / f'gdi_{year}.csv'}")
```

```
# Export temporal analysis
temporal_output_cols = ['code_iris', 'libelle_iris',
                        'GDI_2013', 'GDI_2017', 'GDI_2021',
                        'class_2013', 'class_2017', 'class_2021',
                        'delta_1', 'delta_2', 'delta_total', 'trajectory']

temporal_data[temporal_output_cols].to_csv(TABLES_DIR / 'gdi_temporal_analysis.
 ↪csv', index=False)
print(f"  Exported: {TABLES_DIR / 'gdi_temporal_analysis.csv'}")

# Export geographic data with GDI for mapping in QGIS/other tools
temporal_gdf.to_file(MAPS_DIR / 'gdi_paris_2013_2021.geojson', driver='GeoJSON')
print(f"  Exported: {MAPS_DIR / 'gdi_paris_2013_2021.geojson'}")

print("\n" + "="*60)
print("ALL OUTPUTS EXPORTED SUCCESSFULLY")
print("="*60)
```

```
 Exported: outputs/tables_gdi/gdi_2013.csv
 Exported: outputs/tables_gdi/gdi_2017.csv
 Exported: outputs/tables_gdi/gdi_2021.csv
 Exported: outputs/tables_gdi/gdi_temporal_analysis.csv
 Exported: outputs/maps_gdi/gdi_paris_2013_2021.geojson

============================================================
ALL OUTPUTS EXPORTED SUCCESSFULLY
============================================================
```

1.11 10. Summary Statistics and Insights

1.11.1 Key Findings

[28]:
```
print("="*80)
print("GENTRIFICATION DEGREE INDEX (GDI) - SUMMARY REPORT")
print("Paris Intra-Muros IRIS Analysis (2013-2021)")
print("="*80)

print("\n1. TEMPORAL EVOLUTION OF GDI")
print("-" * 60)
for year, data in [(2013, data_2013_class), (2017, data_2017_class), (2021,␣
 ↪data_2021_class)]:
    mean_gdi = data['GDI'].mean()
    std_gdi = data['GDI'].std()
    print(f"  {year}: Mean = {mean_gdi:+.3f}, Std = {std_gdi:.3f}")

print("\n2. GENTRIFICATION CLASSES (2021)")
print("-" * 60)
class_2021 = data_2021_class['GDI_class'].value_counts().sort_index()
```

```
for cls, count in class_2021.items():
    pct = count / len(data_2021_class) * 100
    print(f"  {cls:25s}: {count:3d} IRIS ({pct:5.1f}%)")

print("\n3. GENTRIFICATION TRAJECTORIES (2013-2021)")
print("-" * 60)
traj_summary = temporal_data.groupby('trajectory').agg({
    'code_iris': 'count',
    'delta_total': 'mean',
    'GDI_2013': 'mean',
    'GDI_2021': 'mean'
}).rename(columns={'code_iris': 'Count'})

for traj in traj_summary.index:
    row = traj_summary.loc[traj]
    pct = row['Count'] / len(temporal_data) * 100
    print(f"\n  {traj}:")
    print(f"    IRIS count:          {int(row['Count']):3d} ({pct:5.1f}%)")
    print(f"    Mean GDI change:     {row['delta_total']:+.3f}")
    print(f"    Mean GDI 2013:       {row['GDI_2013']:+.3f}")
    print(f"    Mean GDI 2021:       {row['GDI_2021']:+.3f}")

print("\n4. TOP 10 MOST GENTRIFIED IRIS (2021)")
print("-" * 60)
top_10 = data_2021_class.nlargest(10, 'GDI')[['code_iris', 'libelle_iris',␣
 ↪'GDI', 'GDI_class']]
for idx, row in top_10.iterrows():
    print(f"  {row['libelle_iris']:40s} GDI={row['GDI']:+.3f}␣
 ↪({row['GDI_class']})")

print("\n5. TOP 10 LEAST GENTRIFIED IRIS (2021)")
print("-" * 60)
bottom_10 = data_2021_class.nsmallest(10, 'GDI')[['code_iris', 'libelle_iris',␣
 ↪'GDI', 'GDI_class']]
for idx, row in bottom_10.iterrows():
    print(f"  {row['libelle_iris']:40s} GDI={row['GDI']:+.3f}␣
 ↪({row['GDI_class']})")

print("\n6. TOP 10 INTENSIFYING IRIS (Largest GDI Increase)")
print("-" * 60)
intensifying_top = temporal_data.nlargest(10, 'delta_total')[['code_iris',␣
 ↪'libelle_iris', 'delta_total', 'GDI_2013', 'GDI_2021']]
for idx, row in intensifying_top.iterrows():
    print(f"  {row['libelle_iris']:40s} Δ={row['delta_total']:+.3f}␣
 ↪({row['GDI_2013']:+.2f}→{row['GDI_2021']:+.2f})")
```

```python
print("\n" + "="*80)
print("END OF REPORT")
print("="*80)
```

```
================================================================================
GENTRIFICATION DEGREE INDEX (GDI) - SUMMARY REPORT
Paris Intra-Muros IRIS Analysis (2013-2021)
================================================================================

1. TEMPORAL EVOLUTION OF GDI
------------------------------------------------------------
  2013: Mean = -0.001, Std = 0.600
  2017: Mean = -0.000, Std = 0.662
  2021: Mean = +0.001, Std = 0.653

2. GENTRIFICATION CLASSES (2021)
------------------------------------------------------------
  Low                        : 213 IRIS ( 24.7%)
  Lower-Intermediate         : 213 IRIS ( 24.7%)
  Upper-Intermediate         : 213 IRIS ( 24.7%)
  High                       : 213 IRIS ( 24.7%)

3. GENTRIFICATION TRAJECTORIES (2013-2021)
------------------------------------------------------------


  Declining:
    IRIS count:          20 (  2.3%)
    Mean GDI change:    -0.656
    Mean GDI 2013:      -0.204
    Mean GDI 2021:      -0.860

  Intensifying:
    IRIS count:          14 (  1.6%)
    Mean GDI change:    +0.588
    Mean GDI 2013:      -0.452
    Mean GDI 2021:      +0.136

  Stable:
    IRIS count:         827 ( 96.1%)
    Mean GDI change:    +0.005
    Mean GDI 2013:      +0.012
    Mean GDI 2021:      +0.019

4. TOP 10 MOST GENTRIFIED IRIS (2021)
------------------------------------------------------------
  Clignancourt 14                          GDI=+1.308 (High)
  Batignolles 1                            GDI=+1.273 (High)
  Folie Méricourt 15                       GDI=+1.211 (High)
  Épinettes 1                              GDI=+1.198 (High)
  Bonne Nouvelle 4                         GDI=+1.185 (High)
  Batignolles 11                           GDI=+1.174 (High)
  Batignolles 6                            GDI=+1.160 (High)
  Arts et Métiers 2                        GDI=+1.143 (High)
  Batignolles 4                            GDI=+1.137 (High)
  Clignancourt 12                          GDI=+1.136 (High)

5. TOP 10 LEAST GENTRIFIED IRIS (2021)
------------------------------------------------------------
  Grandes Carrières 27                     GDI=-2.226 (Low)
  Grandes Carrières 28                     GDI=-2.123 (Low)
  Charonne 9                               GDI=-2.048 (Low)
  Charonne 11                              GDI=-2.045 (Low)
  Saint-Fargeau 3                          GDI=-2.016 (Low)
  Plaisance 3                              GDI=-2.004 (Low)
  Gare 27                                  GDI=-1.999 (Low)
  Amérique 9                               GDI=-1.977 (Low)
  Chapelle 9                               GDI=-1.958 (Low)
  Amérique 12                              GDI=-1.942 (Low)

6. TOP 10 INTENSIFYING IRIS (Largest GDI Increase)
------------------------------------------------------------
  Belleville 9                             Δ=+0.921 (-0.89→+0.04)
  Porte Dauphine 9                         Δ=+0.678 (-0.47→+0.21)
  Goutte d'Or 4                            Δ=+0.670 (-0.21→+0.46)
  Amérique 4                               Δ=+0.635 (-1.62→-0.99)
  Maison Blanche 19                        Δ=+0.631 (-1.70→-1.07)
  Combat 9                                 Δ=+0.594 (+0.00→+0.59)
  Plaine Monceau 7                         Δ=+0.591 (+0.32→+0.91)
  Belleville 4                             Δ=+0.591 (-0.25→+0.34)
  Folie Méricourt 13                       Δ=+0.578 (+0.18→+0.76)
  Saint-Lambert 21                         Δ=+0.568 (-0.81→-0.24)

================================================================================
END OF REPORT
================================================================================
```

1.12 11. Academic Interpretation

1.12.1 Theoretical Implications

The GDI analysis reveals several key patterns consistent with gentrification theory:

1. **Spatial Concentration**: Gentrification is not evenly distributed but concentrated in specific clusters, supporting the "frontier" metaphor (Smith, 1996)
2. **Multi-dimensional Process**: The composite index shows that gentrification operates through multiple channels simultaneously:

    - Economic (income rise)
    - Social (class replacement)
    - Demographic (age shifts)
    - Structural (income source changes)
3. **Temporal Dynamics**: The intensifying/stable/declining classification distinguishes:
    - Active gentrification fronts (intensifying areas)
    - Established elite enclaves (high but stable)
    - Resistant working-class areas (low and stable)
4. **Policy Relevance**: Identification of intensifying areas enables proactive anti-displacement measures

1.12.2 Limitations

- **Displacement not directly measured**: GDI shows compositional change but cannot distinguish replacement vs. uplift
- **Aggregation effects**: IRIS-level analysis may mask block-level heterogeneity

- **Cultural dimensions**: Index focuses on measurable socio-economic indicators, missing cultural/symbolic aspects
- **Causality**: GDI identifies correlation patterns but doesn't establish causal mechanisms

1.12.3 Future Research Directions

1. Integrate DVF (real estate transaction) data to correlate GDI with property value changes
2. Add SIRENE (business) data to examine commercial gentrification
3. Spatial autocorrelation analysis (Moran's I) to identify gentrification clusters
4. Longitudinal tracking of individual IRIS trajectories with qualitative case studies

---

1.13 References

Methodology adapted from: - Glass, R. (1964). *Introduction: Aspects of Change.* London: MacGibbon & Kee. - Smith, N. (1996). *The New Urban Frontier: Gentrification and the Revanchist City.* New York: Routledge. - Hamnett, C. (2003). Gentrification and the middle-class remaking of Inner London, 1961–2001. *Urban Studies*, 40(12), 2401–2426. - Clerval, A. (2013). *Paris sans le peuple: La gentrification de la capitale.* Paris: La Découverte. - Freeman, L. (2005). Displacement or succession? Residential mobility in gentrifying neighborhoods. *Urban Affairs Review*, 40(4), 463–491.

Data sources: - INSEE FiLoSoFi (Fichier Localisé Social et Fiscal) - INSEE Census (Recensement de la Population) - IGN IRIS geographic boundaries

---

Notebook created: 2025-10-16
Version: 4.0
Author: Paris Gentrification Research