

04-environmental-justice-health

November 29, 2025

1 Environmental Justice & Health Equity Analysis

1.1 Executive Summary

This notebook analyzes **environmental and health vulnerabilities** using the **KRL Suite** to combine census demographics with NOAA climate data, FDA health indicators, and USGS environmental data.

1.1.1 KRL Suite Components Used

- `krl_data_connectors.community`: NOAAClimateConnector, USGSConnector, FDACConnector, CensusACSPublicConnector
- `krl_geospatial`: Spatial analysis and clustering
- `krl_core`: Logging utilities

1.1.2 Key Intelligence Questions

1. Which states face the highest environmental burden (climate hazards)?
2. How do environmental conditions correlate with demographic vulnerability?
3. Which regions show compounding risk factors?
4. What is the geographic distribution of environmental justice concerns?

1.1.3 Methodology

NOAA Climate
Hazard Data

Census ACS
Demographics

USGS
Environmental

Environmental
Burden Index

Priority
Intervention Zones

Estimated Time: 20-25 minutes

Difficulty: Intermediate

1.2 1. Environment Setup

```
[7]: # Core imports
import os
import sys
import warnings
from datetime import datetime
from pathlib import Path
import importlib

# Add KRL package paths (handles spaces in path correctly)
_krl_base = os.path.expanduser("~/Documents/GitHub/KRL/Private IP")
for _pkg in ["krl-open-core/src", "krl-data-connectors/src"]:
    _path = os.path.join(_krl_base, _pkg)
    if _path not in sys.path:
        sys.path.insert(0, _path)

# Load environment variables from .env file
from dotenv import load_dotenv
_env_path = os.path.expanduser("~/Documents/GitHub/KRL/Private IP/krl-tutorials/
˓.env")
load_dotenv(_env_path)

# Force complete reload of KRL modules to pick up any changes
_modules_to_reload = [k for k in sys.modules.keys() if k.
˓startswith(('krl_core', 'krl_data_connectors'))]
for _mod in _modules_to_reload:
    del sys.modules[_mod]

import numpy as np
import pandas as pd
from scipy import stats

# Visualization
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots

# =====
# KRL Suite Imports - REAL package imports
# =====

# KRL Data Connectors - Community Tier
```

```

from krl_data_connectors.community import (
    CensusACSPublicConnector, # Demographics
    NOAAClimateConnector, # Climate/weather data
    USGSConnector, # Geological/environmental
    FDACConnector, # Health/food safety
)

# KRL Core
from krl_core import get_logger

warnings.filterwarnings('ignore', category=FutureWarning)

# Initialize logger
logger = get_logger("EnvironmentalJustice")

print("=" * 65)
print(" Environmental Justice & Health Equity Analysis")
print("=" * 65)
print(f" Execution Time: {datetime.now().strftime('%Y-%m-%d %H:%M:%S')}")
print(f" Using KRL Suite (Community Tier)")
print(f" FRED API Key: {'Loaded' if os.getenv('FRED_API_KEY') else 'Not' +
     'found'}")
print("=" * 65)

```

```
=====
Environmental Justice & Health Equity Analysis
=====
```

```
Execution Time: 2025-11-28 04:27:36
Using KRL Suite (Community Tier)
FRED API Key: Loaded
=====
```

```
[8]: # =====
# Initialize KRL Data Connectors
# =====

# Get NOAA API key from environment
noaa_api_key = os.getenv('NOAA_CDO_TOKEN') or os.getenv('NOAA_API_KEY')

# Initialize connectors
census = CensusACSPublicConnector()
noaa = NOAAClimateConnector(api_key=noaa_api_key)
usgs = USGSConnector()

# Test connections
print(" Testing API Connections...")
print(f" Census: {census.connect()}")

```

```

print(f"    NOAA: {noaa.connect()}")
print(f"    USGS: {usgs.connect()}")

print("\n Available Connectors for Environmental Justice Analysis:")
print("    • CensusACSPublicConnector: Demographics & vulnerability indicators")
print("    • NOAAClimateConnector: Temperature extremes, precipitation, storms")
print("    • USGSConnector: Water quality, geological hazards")

```

```

{"timestamp": "2025-11-28T09:27:39.936569Z", "level": "INFO", "name": "CensusACSPublicConnector", "message": "Connector initialized", "source": {"file": "base_connector.py", "line": 81, "function": "__init__"}, "levelname": "INFO", "taskName": "Task-54", "connector": "CensusACSPublicConnector", "cache_dir": "/Users/bcdelo/.krl_cache/censusacspublicconnector", "cache_ttl": 3600, "has_api_key": true}
{"timestamp": "2025-11-28T09:27:39.936938Z", "level": "INFO", "name": "CensusACSPublicConnector", "message": "Initialized Census ACS Public connector (Community tier)", "source": {"file": "census_acs_public.py", "line": 101, "function": "__init__"}, "levelname": "INFO", "taskName": "Task-54", "geography": "state-level only"}
{"timestamp": "2025-11-28T09:27:39.938537Z", "level": "INFO", "name": "NOAAClimateConnector", "message": "Connector initialized", "source": {"file": "base_connector.py", "line": 81, "function": "__init__"}, "levelname": "INFO", "taskName": "Task-54", "connector": "NOAAClimateConnector", "cache_dir": "/Users/bcdelo/.krl_cache/noaaclimateconnector", "cache_ttl": 3600, "has_api_key": true}
{"timestamp": "2025-11-28T09:27:39.939318Z", "level": "WARNING", "name": "USGSConnector", "message": "No API key provided", "source": {"file": "base_connector.py", "line": 74, "function": "__init__"}, "levelname": "WARNING", "taskName": "Task-54", "connector": "USGSConnector"}
{"timestamp": "2025-11-28T09:27:39.939505Z", "level": "INFO", "name": "USGSConnector", "message": "Connector initialized", "source": {"file": "base_connector.py", "line": 81, "function": "__init__"}, "levelname": "INFO", "taskName": "Task-54", "connector": "USGSConnector", "cache_dir": "/Users/bcdelo/.krl_cache/usgsconnector", "cache_ttl": 3600, "has_api_key": false}
{"timestamp": "2025-11-28T09:27:39.936938Z", "level": "INFO", "name": "CensusACSPublicConnector", "message": "Initialized Census ACS Public connector (Community tier)", "source": {"file": "census_acs_public.py", "line": 101, "function": "__init__"}, "levelname": "INFO", "taskName": "Task-54", "geography": "state-level only"}
{"timestamp": "2025-11-28T09:27:39.938537Z", "level": "INFO", "name": "NOAAClimateConnector", "message": "Connector initialized", "source": {"file": "base_connector.py", "line": 81, "function": "__init__"}, "levelname": "INFO", "taskName": "Task-54", "connector": "NOAAClimateConnector", "cache_dir": "/Users/bcdelo/.krl_cache/noaaclimateconnector", "cache_ttl": 3600, "has_api_key": true}
{"timestamp": "2025-11-28T09:27:39.939318Z", "level": "WARNING", "name": "USGSConnector", "message": "No API key provided", "source": {"file": "base_connector.py", "line": 74, "function": "__init__"}, "levelname": "WARNING", "taskName": "Task-54", "connector": "USGSConnector"}

```

```

"base_connector.py", "line": 74, "function": "__init__"}, "levelname": "WARNING", "taskName": "Task-54", "connector": "USGSConnector"}
{"timestamp": "2025-11-28T09:27:39.939505Z", "level": "INFO", "name": "USGSConnector", "message": "Connector initialized", "source": {"file": "base_connector.py", "line": 81, "function": "__init__"}, "levelname": "INFO", "taskName": "Task-54", "connector": "USGSConnector", "cache_dir": "/Users/bcdelo/.krl_cache/usgsconnector", "cache_ttl": 3600, "has_api_key": false}
    Testing API Connections...
    Testing API Connections...
{"timestamp": "2025-11-28T09:27:40.258806Z", "level": "INFO", "name": "CensusACSPublicConnector", "message": "Successfully connected to Census API", "source": {"file": "census_acs_public.py", "line": 137, "function": "connect"}, "levelname": "INFO", "taskName": "Task-54"}
{"timestamp": "2025-11-28T09:27:40.258806Z", "level": "INFO", "name": "CensusACSPublicConnector", "message": "Successfully connected to Census API", "source": {"file": "census_acs_public.py", "line": 137, "function": "connect"}, "levelname": "INFO", "taskName": "Task-54"}
    Census: True
    Census: True
{"timestamp": "2025-11-28T09:27:40.522844Z", "level": "INFO", "name": "NOAAClimateConnector", "message": "Successfully connected to NOAA CDO API", "source": {"file": "noaa_climate_current.py", "line": 189, "function": "connect"}, "levelname": "INFO", "taskName": "Task-54"}
    NOAA: None
{"timestamp": "2025-11-28T09:27:40.523864Z", "level": "INFO", "name": "USGSConnector", "message": "Successfully connected to USGS data sources", "source": {"file": "usgs_earthquakes.py", "line": 138, "function": "connect"}, "levelname": "INFO", "taskName": "Task-54"}
    USGS: None

```

Available Connectors for Environmental Justice Analysis:

- CensusACSPublicConnector: Demographics & vulnerability indicators
- NOAAClimateConnector: Temperature extremes, precipitation, storms
- USGSConnector: Water quality, geological hazards

```

{"timestamp": "2025-11-28T09:27:40.522844Z", "level": "INFO", "name": "NOAAClimateConnector", "message": "Successfully connected to NOAA CDO API", "source": {"file": "noaa_climate_current.py", "line": 189, "function": "connect"}, "levelname": "INFO", "taskName": "Task-54"}
    NOAA: None
{"timestamp": "2025-11-28T09:27:40.523864Z", "level": "INFO", "name": "USGSConnector", "message": "Successfully connected to USGS data sources", "source": {"file": "usgs_earthquakes.py", "line": 138, "function": "connect"}, "levelname": "INFO", "taskName": "Task-54"}
    USGS: None

```

Available Connectors for Environmental Justice Analysis:

- CensusACSPublicConnector: Demographics & vulnerability indicators

- NOAAClimateConnector: Temperature extremes, precipitation, storms
- USGSConnector: Water quality, geological hazards

1.3 2. Data Collection: Environmental & Demographic Data

We'll combine multiple data sources to build a comprehensive environmental justice index.

```
[9]: # =====
# Fetch Demographics: Vulnerable Populations
# =====

# Get state-level demographics focusing on vulnerability indicators
demographics = census.get_demographics_by_state(year=2022)

# Rename columns for clarity
demographics_renamed = demographics.rename(columns={
    'B01001_001E': 'population',
    'B01002_001E': 'median_age',
    'B19013_001E': 'median_income',
    'B17001_002E': 'poverty_pop',
    'B02001_002E': 'white_pop',
    'B02001_003E': 'black_pop',
    'B02001_005E': 'asian_pop',
    'B03003_003E': 'hispanic_pop',
})

# Compute vulnerability metrics
demographics_renamed['poverty_rate'] = (demographics_renamed['poverty_pop'] / demographics_renamed['population']) * 100
demographics_renamed['minority_pct'] = ((demographics_renamed['population'] - demographics_renamed['white_pop']) / demographics_renamed['population']) * 100

print(" Demographic Vulnerability Indicators Retrieved:")
print(f" States: {len(demographics_renamed)}")
print(f" Variables: population, median_income, poverty_rate, minority_pct")

demographics_renamed[['NAME', 'population', 'median_income', 'poverty_rate', 'minority_pct']].head(10)

{
  "timestamp": "2025-11-28T09:27:44.184379Z", "level": "INFO", "name": "CensusACSPublicConnector", "message": "Fetching Census ACS data for 2022", "source": {"file": "census_acs_public.py", "line": 175, "function": "get_data"}, "levelname": "INFO", "taskName": "Task-57", "year": 2022, "variables": 9, "geography": "state"}
{
  "timestamp": "2025-11-28T09:27:44.460374Z", "level": "INFO", "name": "CensusACSPublicConnector", "message": "Retrieved data for 52 states", "source": "CensusACSPublicConnector"
}
```

```

{"file": "census_acs_public.py", "line": 197, "function": "get_data"},  

"levelname": "INFO", "taskName": "Task-57", "year": 2022, "rows": 52}  

{"timestamp": "2025-11-28T09:27:44.460374Z", "level": "INFO", "name":  

"CensusACSPublicConnector", "message": "Retrieved data for 52 states", "source":  

{"file": "census_acs_public.py", "line": 197, "function": "get_data"},  

"levelname": "INFO", "taskName": "Task-57", "year": 2022, "rows": 52}  

    Demographic Vulnerability Indicators Retrieved:  

        States: 52  

        Variables: population, median_income, poverty_rate, minority_pct  

    Demographic Vulnerability Indicators Retrieved:  

        States: 52  

        Variables: population, median_income, poverty_rate, minority_pct

```

[9]:

	NAME	population	median_income	poverty_rate	minority_pct
0	Alabama	5028092	59609	15.292023	33.791744
1	Alaska	734821	86370	10.237459	38.696363
2	Arizona	7172282	72581	12.783602	33.330814
3	Arkansas	3018669	56335	15.759562	27.340560
4	California	39356104	91905	11.904817	51.866018
5	Colorado	5770790	87598	9.359290	23.868153
6	Connecticut	3611317	90213	9.849371	30.159385
7	Delaware	993635	79325	10.848048	36.169318
8	District of Columbia	670587	101722	14.619878	60.387988
9	Florida	21634529	67917	12.598532	36.178828

1.4 3. Environmental Data from NOAA

NOAA Climate data provides indicators of environmental hazards and climate vulnerability.

[10]:

```

# =====
# Create Environmental Burden Simulation
# (NOAA API provides climate data; we simulate EJ indicators for demo)
# =====

# Note: Full environmental justice data requires Professional tier
# with EJSscreen API access. Here we create illustrative metrics.

np.random.seed(42)
n_states = len(demographics_renamed)

# Simulate environmental burden indicators (would come from NOAA/EPA in ↴
# production)
env_data = pd.DataFrame({
    'state': demographics_renamed['NAME'].values,
    # Climate hazard scores (simulated based on regional patterns)
    'heat_risk': np.random.uniform(40, 90, n_states),
    'flood_risk': np.random.uniform(30, 80, n_states),
    'air_quality_index': np.random.uniform(30, 120, n_states),
})

```

```

        'water_contamination_risk': np.random.uniform(10, 60, n_states),
    })

# Compute composite environmental burden
env_data['environmental_burden'] = (
    env_data['heat_risk'].rank(pct=True) * 0.3 +
    env_data['flood_risk'].rank(pct=True) * 0.3 +
    env_data['air_quality_index'].rank(pct=True) * 0.2 +
    env_data['water_contamination_risk'].rank(pct=True) * 0.2
) * 100

print(" Environmental Burden Indicators (Simulated for Demo):")
print("     Note: Production would use NOAA/EPA APIs via Professional tier")
env_data.head(10)

```

Environmental Burden Indicators (Simulated for Demo):

Note: Production would use NOAA/EPA APIs via Professional tier

```
[10]:      state  heat_risk  flood_risk  air_quality_index \
0       Alabama  58.727006   76.974947   111.680983
1        Alaska  87.535715   74.741368   52.436301
2      Arizona  76.599697   59.894999   66.934463
3     Arkansas  69.932924   76.093712   97.999602
4    California  47.800932   34.424625   50.591835
5      Colorado  47.799726   39.799143   36.928192
6  Connecticut  42.904181   32.261364   56.077631
7     Delaware  83.308807   46.266517   44.509916
8 District of Columbia  70.055751   49.433864   113.672789
9       Florida  75.403629   43.567452   102.730834

      water_contamination_risk  environmental_burden
0           43.606777          74.807692
1           48.080981          78.076923
2           21.881877          57.115385
3           46.410817          78.076923
4           28.389157          23.076923
5           41.615292          25.384615
6           41.676486          23.846154
7           36.788734          50.769231
8           14.514489          54.423077
9           51.765125          64.038462
```

```
[11]: # =====
# Merge Demographics with Environmental Data
# =====

# Merge on state name
ej_df = demographics_renamed.merge(env_data, left_on='NAME', right_on='state')
```

```

# Compute Environmental Justice Index
# High EJ concern = high environmental burden + high vulnerability
ej_df['vulnerability_score'] = (
    ej_df['poverty_rate'].rank(pct=True) * 0.5 +
    ej_df['minority_pct'].rank(pct=True) * 0.5
) * 100

ej_df['ej_index'] = (ej_df['environmental_burden'] +_
    ej_df['vulnerability_score']) / 2

# Rank by EJ concern
ej_df = ej_df.sort_values('ej_index', ascending=False)
ej_df['ej_rank'] = range(1, len(ej_df) + 1)

print(" Environmental Justice Index Computed:")
ej_df[['NAME', 'environmental_burden', 'vulnerability_score', 'ej_index',_ 
    'ej_rank']].head(10)

```

Environmental Justice Index Computed:

```
[11]:
```

	NAME	environmental_burden	vulnerability_score	\
51	Puerto Rico	59.230769	98.076923	
43	Texas	70.384615	80.769231	
36	Oklahoma	78.269231	71.153846	
3	Arkansas	78.076923	71.153846	
0	Alabama	74.807692	74.038462	
18	Louisiana	60.000000	87.500000	
8	District of Columbia	54.423077	90.384615	
24	Mississippi	47.500000	92.307692	
35	Ohio	83.653846	52.884615	
9	Florida	64.038462	67.307692	
	ej_index	ej_rank		
51	78.653846	1		
43	75.576923	2		
36	74.711538	3		
3	74.615385	4		
0	74.423077	5		
18	73.750000	6		
8	72.403846	7		
24	69.903846	8		
35	68.269231	9		
9	65.673077	10		

1.5 4. Visualization: Environmental Justice Mapping

```
[12]: # =====
# Visualization: Environmental Justice Index Map
# =====

# Add state abbreviations
state_abbrev = {
    'Alabama': 'AL', 'Alaska': 'AK', 'Arizona': 'AZ', 'Arkansas': 'AR',
    'California': 'CA',
    'Colorado': 'CO', 'Connecticut': 'CT', 'Delaware': 'DE', 'Florida': 'FL',
    'Georgia': 'GA',
    'Hawaii': 'HI', 'Idaho': 'ID', 'Illinois': 'IL', 'Indiana': 'IN', 'Iowa': 'IA',
    'Kansas': 'KS', 'Kentucky': 'KY', 'Louisiana': 'LA', 'Maine': 'ME',
    'Maryland': 'MD',
    'Massachusetts': 'MA', 'Michigan': 'MI', 'Minnesota': 'MN', 'Mississippi': 'MS',
    'Missouri': 'MO',
    'Montana': 'MT', 'Nebraska': 'NE', 'Nevada': 'NV', 'New Hampshire': 'NH',
    'New Jersey': 'NJ',
    'New Mexico': 'NM', 'New York': 'NY', 'North Carolina': 'NC', 'North Dakota': 'ND',
    'Ohio': 'OH',
    'Oklahoma': 'OK', 'Oregon': 'OR', 'Pennsylvania': 'PA', 'Rhode Island': 'RI',
    'South Carolina': 'SC',
    'South Dakota': 'SD', 'Tennessee': 'TN', 'Texas': 'TX', 'Utah': 'UT',
    'Vermont': 'VT',
    'Virginia': 'VA', 'Washington': 'WA', 'West Virginia': 'WV', 'Wisconsin': 'WI',
    'Wyoming': 'WY',
    'District of Columbia': 'DC', 'Puerto Rico': 'PR'
}

ej_df['state_abbrev'] = ej_df['NAME'].map(state_abbrev)

fig = px.choropleth(
    ej_df,
    locations='state_abbrev',
    locationmode='USA-states',
    color='ej_index',
    scope='usa',
    color_continuous_scale='RdYlGn_r',
    hover_name='NAME',
    hover_data={
        'environmental_burden': ':.1f',
        'vulnerability_score': ':.1f',
        'ej_index': ':.1f',
        'poverty_rate': ':.1f'
    },
}
```

```

        title='Environmental Justice Index by State',
    )

fig.update_layout(height=500)
fig.show()

```

1.6 5. Disparity Analysis: Environmental Burden vs Demographics

```
[13]: # =====
# Scatter: Environmental Burden vs Vulnerability
# =====

fig = px.scatter(
    ej_df,
    x='vulnerability_score',
    y='environmental_burden',
    color='ej_index',
    size='population',
    hover_name='NAME',
    title='Environmental Burden vs Community Vulnerability',
    labels={
        'vulnerability_score': 'Vulnerability Score (Poverty + Minority %)',
        'environmental_burden': 'Environmental Burden Score',
        'ej_index': 'EJ Index'
    },
    color_continuous_scale='RdYlGn_r',
    template='plotly_white',
)

# Add quadrant lines
fig.add_vline(x=50, line_dash="dash", line_color="gray", opacity=0.5)
fig.add_hline(y=50, line_dash="dash", line_color="gray", opacity=0.5)

# Label quadrants
fig.add_annotation(x=75, y=75, text="HIGH PRIORITY", showarrow=False, ↴
    font=dict(color='red'))
fig.add_annotation(x=25, y=25, text="Lower concern", showarrow=False, ↴
    font=dict(color='green'))

fig.update_layout(height=550)
fig.show()
```

```
[14]: # =====
# Correlation Analysis
# =====

# Compute correlations
```

```

corr_matrix = ej_df[['poverty_rate', 'minority_pct', 'environmental_burden', 'vulnerability_score', 'ej_index']].corr()

print(" Correlation Matrix:")
print(corr_matrix.round(3))

# Key correlations
print(f"\n Key Correlations:")
print(f" • Poverty Env Burden: {corr_matrix.loc['poverty_rate', 'environmental_burden']:.3f}")
print(f" • Minority % Env Burden: {corr_matrix.loc['minority_pct', 'environmental_burden']:.3f}")
print(f" • Vulnerability Env Burden: {corr_matrix.loc['vulnerability_score', 'environmental_burden']:.3f}")

```

Correlation Matrix:

	poverty_rate	minority_pct	environmental_burden
poverty_rate	1.000	0.331	0.188
minority_pct	0.331	1.000	0.138
environmental_burden	0.188	0.138	1.000
vulnerability_score	0.635	0.735	0.262
ej_index	0.572	0.624	0.682

	vulnerability_score	ej_index
poverty_rate	0.635	0.572
minority_pct	0.735	0.624
environmental_burden	0.262	0.682
vulnerability_score	1.000	0.884
ej_index	0.884	1.000

Key Correlations:

- Poverty Env Burden: 0.188
- Minority % Env Burden: 0.138
- Vulnerability Env Burden: 0.262

1.7 6. Priority Intervention Zones

```
[15]: # =====
# Classify Priority Intervention Zones
# =====

def classify_ej_priority(row):
    if row['environmental_burden'] > 60 and row['vulnerability_score'] > 60:
        return 'Critical'
    elif row['environmental_burden'] > 50 or row['vulnerability_score'] > 60:
        return 'High'
    elif row['environmental_burden'] > 40 or row['vulnerability_score'] > 50:
        return 'Medium'
    else:
        return 'Low'
```

```

        return 'Moderate'
    else:
        return 'Lower'

ej_df['priority'] = ej_df.apply(classify_ej_priority, axis=1)

# Summary by priority
priority_summary = ej_df.groupby('priority').agg({
    'NAME': 'count',
    'population': 'sum',
    'ej_index': 'mean',
}).round(1)
priority_summary.columns = ['States', 'Total Population', 'Avg EJ Index']

print(" Environmental Justice Priority Classification:")
priority_summary

```

Environmental Justice Priority Classification:

```
[15]:      States  Total Population  Avg EJ Index
priority
Critical      5          62895129       73.0
High          27         167285750       57.6
Lower          9          37497767       33.1
Moderate      11         66691329       39.3
```

```
[16]: # List critical and high priority states
critical = ej_df[ej_df['priority'] == 'Critical']
high = ej_df[ej_df['priority'] == 'High']

print(" CRITICAL PRIORITY STATES:")
for _, row in critical.iterrows():
    print(f"  • {row['NAME']}: EJ Index {row['ej_index']:.1f}")

print(f"\n HIGH PRIORITY STATES ({len(high)}):")
for _, row in high.head(5).iterrows():
    print(f"  • {row['NAME']}: EJ Index {row['ej_index']:.1f}")
```

CRITICAL PRIORITY STATES:

- Texas: EJ Index 75.6
- Oklahoma: EJ Index 74.7
- Arkansas: EJ Index 74.6
- Alabama: EJ Index 74.4
- Florida: EJ Index 65.7

HIGH PRIORITY STATES (27):

- Puerto Rico: EJ Index 78.7
- Louisiana: EJ Index 73.8

- District of Columbia: EJ Index 72.4
- Mississippi: EJ Index 69.9
- Ohio: EJ Index 68.3

1.8 7. Key Insights & Policy Recommendations

```
[17]: # =====
# Key Insights Summary
# =====

print("==" * 65)
print(" KEY INSIGHTS: Environmental Justice Analysis")
print("==" * 65)

print(f"\n Geographic Scope: 50 States + DC")
print(f" Demographics: 2022 Census ACS")

print(f"\n ENVIRONMENTAL BURDEN SUMMARY:")
print(f"    • Avg Environmental Burden: {ej_df['environmental_burden'].mean():.2f}")
print(f"    • Max Environmental Burden: {ej_df['environmental_burden'].max():.2f}")
print(f"    • States with High Burden (>60): {(ej_df['environmental_burden'] > 60).sum()}")


print(f"\n VULNERABILITY SUMMARY:")
print(f"    • Avg Vulnerability Score: {ej_df['vulnerability_score'].mean():.2f}")
print(f"    • States with High Vulnerability (>60):{(ej_df['vulnerability_score'] > 60).sum()}")


print(f"\n EJ PRIORITY ZONES:")
print(f"    • Critical: {(ej_df['priority'] == 'Critical').sum()} states")
print(f"    • High: {(ej_df['priority'] == 'High').sum()} states")

print("\n" + "==" * 65)
print(" POLICY RECOMMENDATIONS")
print("==" * 65)
print("""
1. CUMULATIVE IMPACT: States with both high environmental burden
AND high vulnerability need priority intervention.

2. TARGETED RESOURCES: Direct EPA/HHS resources to high-EJ areas.

3. HEALTH MONITORING: Establish enhanced health surveillance in
critical priority zones.
""")
```

4. GRANULAR ANALYSIS: Upgrade to Professional tier for tract-level EJScreen and CDC PLACES data.
****)

=====
KEY INSIGHTS: Environmental Justice Analysis
=====

Geographic Scope: 50 States + DC

Demographics: 2022 Census ACS

ENVIRONMENTAL BURDEN SUMMARY:

- Avg Environmental Burden: 51.0
- Max Environmental Burden: 83.7
- States with High Burden (>60): 13

VULNERABILITY SUMMARY:

- Avg Vulnerability Score: 51.0
- States with High Vulnerability (>60): 18

EJ PRIORITY ZONES:

- Critical: 5 states
- High: 27 states

=====
POLICY RECOMMENDATIONS
=====

1. CUMULATIVE IMPACT: States with both high environmental burden AND high vulnerability need priority intervention.
2. TARGETED RESOURCES: Direct EPA/HHS resources to high-EJ areas.
3. HEALTH MONITORING: Establish enhanced health surveillance in critical priority zones.
4. GRANULAR ANALYSIS: Upgrade to Professional tier for tract-level EJScreen and CDC PLACES data.

1.9 8. Data Provenance & Next Steps

[18]: # ======
Session Information
======

import sys

```

print(" Session Information")
print("==" * 50)
print(f"Python Version: {sys.version}")
print(f"Pandas Version: {pd.__version__}")
print(f"NumPy Version: {np.__version__}")
print()
print(" KRL Suite Packages Used:")
print("     • krl_data_connectors.community.CensusACSPublicConnector")
print("     • krl_data_connectors.community.NOAAClimateConnector")
print("     • krl_data_connectors.community.USGSSConnector")
print("     • krl_core (Logging)")
print()
print(" Note: Environmental burden indicators are simulated for demo.")
print(" Production analysis requires Professional tier with EJScreen API.")
print()
print(f" Execution Completed: {datetime.now().isoformat()}")

```

Session Information
=====

Python Version: 3.13.7 (main, Aug 14 2025, 11:12:11) [Clang 17.0.0
(clang-1700.0.13.3)]
Pandas Version: 2.3.3
NumPy Version: 2.3.4

KRL Suite Packages Used:

- krl_data_connectors.community.CensusACSPublicConnector
- krl_data_connectors.community.NOAAClimateConnector
- krl_data_connectors.community.USGSSConnector
- krl_core (Logging)

Note: Environmental burden indicators are simulated for demo.
Production analysis requires Professional tier with EJScreen API.

Execution Completed: 2025-11-28T04:28:19.058095

1.9.1 Next Steps & Professional Features

```

# Professional Tier: EPA EJScreen Integration
from krl_data_connectors.professional import EJScreenConnector, CDCPlacesConnector

ejscreen = EJScreenConnector(license_key="YOUR_KEY")
places = CDCPlacesConnector(license_key="YOUR_KEY")

# Tract-level environmental burden
ej_data = ejscreen.get_tract_data(state="CA", variables=[
    'PM25', 'OZONE', 'DIESEL', 'CANCER', 'RESP', 'LEAD'
])

```

```
# Health outcomes by tract
health_data = places.get_tract_health(state="CA", measures=[
    'DIABETES', 'OBESITY', 'CHD', 'COPD', 'ASTHMA'
])
```

1.9.2 Related Notebooks

- [05-climate-resilience-economics.ipynb](#): Climate risk analysis
 - [10-urban-resilience-dashboard.ipynb](#): Multi-source integration
-

© 2025 KR-Labs. Licensed under CC-BY-4.0.

This notebook is part of the Khipu Socioeconomic Analysis Suite public showcase.