03-economic-mobility-deserts

November 29, 2025

1 Economic Mobility Deserts: Identifying Areas of Declining Opportunity

1.1 Executive Summary

This notebook identifies **economic mobility deserts** - areas where economic opportunity is declining based on real Census and economic data.

1.1.1 KRL Suite Components Used

- **krl_data_connectors.community**: `CensusACSPublicConnector` for demographics, `BEAConnector` for regional income
- **krl_models**: `LocationQuotientModel`, `ShiftShareModel` for regional economic analysis
- **krl_policy**: `TreatmentEffectEstimator`, `DifferenceInDifferences` for causal inference
- **krl_core**: Logging and utilities

1.1.2 Key Intelligence Questions

1. Which states show declining economic opportunity?
2. How do income, poverty, and education levels correlate?
3. What regional economic structures drive opportunity differences?
4. What policy factors correlate with mobility outcomes?

1.1.3 Methodology

```
Census ACS            BEA Regional          BLS Labor
Demographics          Income Data           Statistics






                     Opportunity Index
                     Construction
```

```
Location          Shift-Share          Causal
Quotient          Analysis             Inference
```

Estimated Time: 25-30 minutes
Difficulty: Intermediate to Advanced

1.2 1. Environment Setup and Data Loading

[1]:
```
# 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",
    "krl-model-zoo-v2-2.0.0-dev",  # Use the dev version with all models
    "krl-causal-policy-toolkit/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/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', 'krl_models', 'krl_policy'))]
for _mod in _modules_to_reload:
    del sys.modules[_mod]

import numpy as np
import pandas as pd

# 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,  # Census demographics
    BEAConnector,              # Bureau of Economic Analysis
    FREDBasicConnector,        # Federal Reserve data
    BLSBasicConnector,         # Labor statistics
)

# KRL Models - Regional Analysis
from krl_models import LocationQuotientModel, ShiftShareModel

# KRL Policy - Causal Inference
from krl_policy import TreatmentEffectEstimator, DifferenceInDifferences

# KRL Core - Utilities
from krl_core import get_logger

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

# Initialize logger
logger = get_logger("EconomicMobilityDeserts")

print("=" * 65)
print("  Economic Mobility Deserts Analysis")
print("=" * 65)
print(f"  Execution Time: {datetime.now().strftime('%Y-%m-%d %H:%M:%S')}")
print(f"  Using KRL Suite (Community Tier + Models + Policy)")
print(f"  API Keys: FRED={' ' if os.getenv('FRED_API_KEY') else ' '} | BEA={' '␣
↪if os.getenv('BEA_API_KEY') else ' '} | Census={' ' if os.
↪getenv('CENSUS_API_KEY') else ' '}")
print("=" * 65)
```

```
=================================================================
  Economic Mobility Deserts Analysis
=================================================================
  Execution Time: 2025-11-28 04:19:47
  Using KRL Suite (Community Tier + Models + Policy)
  API Keys: FRED=  | BEA=  | Census=
=================================================================
```

[2]:
```
# =============================================================================
# Initialize KRL Data Connectors
```

```python
# ============================================================================

# Initialize all connectors
census = CensusACSPublicConnector()
bea = BEAConnector()
fred = FREDBasicConnector()
bls = BLSBasicConnector()

# Test connections
print("  Testing API Connections...")
print(f"   Census: {census.connect()}")
print(f"   BEA: {bea.connect()}")
print(f"   FRED: {fred.connect()}")
print(f"   BLS: {bls.connect()}")

print("\n  KRL Suite Packages Loaded:")
print("   • krl_data_connectors: Census, BEA, FRED, BLS")
print("   • krl_models: LocationQuotientModel, ShiftShareModel")
print("   • krl_policy: TreatmentEffectEstimator, DifferenceInDifferences")
```

```
{"timestamp": "2025-11-28T09:19:47.697747Z", "level": "INFO", "name":
"CensusACSPublicConnector", "message": "Connector initialized", "source":
{"file": "base_connector.py", "line": 81, "function": "__init__"}, "levelname":
"INFO", "taskName": "Task-27", "connector": "CensusACSPublicConnector",
"cache_dir": "/Users/bcdelo/.krl_cache/censusacspublicconnector", "cache_ttl":
3600, "has_api_key": true}
{"timestamp": "2025-11-28T09:19:47.698040Z", "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-27",
"geography": "state-level only"}
{"timestamp": "2025-11-28T09:19:47.698859Z", "level": "INFO", "name":
"BEAConnector", "message": "Connector initialized", "source": {"file":
"base_connector.py", "line": 81, "function": "__init__"}, "levelname": "INFO",
"taskName": "Task-27", "connector": "BEAConnector", "cache_dir":
"/Users/bcdelo/.krl_cache/beaconnector", "cache_ttl": 3600, "has_api_key": true}
{"timestamp": "2025-11-28T09:19:47.699031Z", "level": "INFO", "name":
"BEAConnector", "message": "Initialized BEA connector", "source": {"file":
"bea_national.py", "line": 93, "function": "__init__"}, "levelname": "INFO",
"taskName": "Task-27"}
{"timestamp": "2025-11-28T09:19:47.699818Z", "level": "INFO", "name":
"FREDBasicConnector", "message": "Connector initialized", "source": {"file":
"base_connector.py", "line": 81, "function": "__init__"}, "levelname": "INFO",
"taskName": "Task-27", "connector": "FREDBasicConnector", "cache_dir":
"/Users/bcdelo/.krl_cache/fredbasicconnector", "cache_ttl": 3600, "has_api_key":
true}
{"timestamp": "2025-11-28T09:19:47.700011Z", "level": "INFO", "name":
"FREDBasicConnector", "message": "Initialized FRED Basic connector (Community
tier)", "source": {"file": "fred_basic.py", "line": 96, "function": "__init__"},
"levelname": "INFO", "taskName": "Task-27", "available_series": 15}
{"timestamp": "2025-11-28T09:19:47.700680Z", "level": "WARNING", "name":
"BLSBasicConnector", "message": "No API key provided", "source": {"file":
"base_connector.py", "line": 74, "function": "__init__"}, "levelname":
"WARNING", "taskName": "Task-27", "connector": "BLSBasicConnector"}
{"timestamp": "2025-11-28T09:19:47.700906Z", "level": "INFO", "name":
"BLSBasicConnector", "message": "Connector initialized", "source": {"file":
"base_connector.py", "line": 81, "function": "__init__"}, "levelname": "INFO",
"taskName": "Task-27", "connector": "BLSBasicConnector", "cache_dir":
"/Users/bcdelo/.krl_cache/blsbasicconnector", "cache_ttl": 3600, "has_api_key":
false}
{"timestamp": "2025-11-28T09:19:47.701150Z", "level": "INFO", "name":
"BLSBasicConnector", "message": "Initialized BLS Basic connector (Community
tier)", "source": {"file": "bls_basic.py", "line": 89, "function": "__init__"},
"levelname": "INFO", "taskName": "Task-27", "available_series": 8}
{"timestamp": "2025-11-28T09:19:47.698040Z", "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-27",
"geography": "state-level only"}
{"timestamp": "2025-11-28T09:19:47.698859Z", "level": "INFO", "name":
"BEAConnector", "message": "Connector initialized", "source": {"file":
"base_connector.py", "line": 81, "function": "__init__"}, "levelname": "INFO",
"taskName": "Task-27", "connector": "BEAConnector", "cache_dir":
"/Users/bcdelo/.krl_cache/beaconnector", "cache_ttl": 3600, "has_api_key": true}
{"timestamp": "2025-11-28T09:19:47.699031Z", "level": "INFO", "name":
"BEAConnector", "message": "Initialized BEA connector", "source": {"file":
"bea_national.py", "line": 93, "function": "__init__"}, "levelname": "INFO",
"taskName": "Task-27"}
{"timestamp": "2025-11-28T09:19:47.699818Z", "level": "INFO", "name":
"FREDBasicConnector", "message": "Connector initialized", "source": {"file":
"base_connector.py", "line": 81, "function": "__init__"}, "levelname": "INFO",
"taskName": "Task-27", "connector": "FREDBasicConnector", "cache_dir":
"/Users/bcdelo/.krl_cache/fredbasicconnector", "cache_ttl": 3600, "has_api_key":
true}
{"timestamp": "2025-11-28T09:19:47.700011Z", "level": "INFO", "name":
"FREDBasicConnector", "message": "Initialized FRED Basic connector (Community
tier)", "source": {"file": "fred_basic.py", "line": 96, "function": "__init__"},
"levelname": "INFO", "taskName": "Task-27", "available_series": 15}
{"timestamp": "2025-11-28T09:19:47.700680Z", "level": "WARNING", "name":
"BLSBasicConnector", "message": "No API key provided", "source": {"file":
"base_connector.py", "line": 74, "function": "__init__"}, "levelname":
"WARNING", "taskName": "Task-27", "connector": "BLSBasicConnector"}
{"timestamp": "2025-11-28T09:19:47.700906Z", "level": "INFO", "name":
"BLSBasicConnector", "message": "Connector initialized", "source": {"file":
"base_connector.py", "line": 81, "function": "__init__"}, "levelname": "INFO",
"taskName": "Task-27", "connector": "BLSBasicConnector", "cache_dir":
```

```
"/Users/bcdelo/.krl_cache/blsbasicconnector", "cache_ttl": 3600, "has_api_key":
false}
{"timestamp": "2025-11-28T09:19:47.701150Z", "level": "INFO", "name":
"BLSBasicConnector", "message": "Initialized BLS Basic connector (Community
tier)", "source": {"file": "bls_basic.py", "line": 89, "function": "__init__"},
"levelname": "INFO", "taskName": "Task-27", "available_series": 8}
 Testing API Connections…
 Testing API Connections…
{"timestamp": "2025-11-28T09:19:48.126164Z", "level": "INFO", "name":
"CensusACSPublicConnector", "message": "Successfully connected to Census API",
"source": {"file": "census_acs_public.py", "line": 137, "function": "connect"},
"levelname": "INFO", "taskName": "Task-27"}
{"timestamp": "2025-11-28T09:19:48.126164Z", "level": "INFO", "name":
"CensusACSPublicConnector", "message": "Successfully connected to Census API",
"source": {"file": "census_acs_public.py", "line": 137, "function": "connect"},
"levelname": "INFO", "taskName": "Task-27"}
   Census: True
   BEA: None
   Census: True
   BEA: None
{"timestamp": "2025-11-28T09:19:48.408866Z", "level": "INFO", "name":
"FREDBasicConnector", "message": "Successfully connected to FRED API", "source":
{"file": "fred_basic.py", "line": 131, "function": "connect"}, "levelname":
"INFO", "taskName": "Task-27"}
{"timestamp": "2025-11-28T09:19:48.408866Z", "level": "INFO", "name":
"FREDBasicConnector", "message": "Successfully connected to FRED API", "source":
{"file": "fred_basic.py", "line": 131, "function": "connect"}, "levelname":
"INFO", "taskName": "Task-27"}
   FRED: True
   FRED: True
{"timestamp": "2025-11-28T09:19:48.831083Z", "level": "INFO", "name":
"BLSBasicConnector", "message": "Successfully connected to BLS API", "source":
{"file": "bls_basic.py", "line": 128, "function": "connect"}, "levelname":
"INFO", "taskName": "Task-27"}
   BLS: True

 KRL Suite Packages Loaded:
   • krl_data_connectors: Census, BEA, FRED, BLS
   • krl_models: LocationQuotientModel, ShiftShareModel
   • krl_policy: TreatmentEffectEstimator, DifferenceInDifferences
{"timestamp": "2025-11-28T09:19:48.831083Z", "level": "INFO", "name":
"BLSBasicConnector", "message": "Successfully connected to BLS API", "source":
{"file": "bls_basic.py", "line": 128, "function": "connect"}, "levelname":
"INFO", "taskName": "Task-27"}
   BLS: True

 KRL Suite Packages Loaded:
   • krl_data_connectors: Census, BEA, FRED, BLS
```

- `krl_models: LocationQuotientModel, ShiftShareModel`
- `krl_policy: TreatmentEffectEstimator, DifferenceInDifferences`

1.3 2. Data Collection: Multi-Source Economic Indicators

We'll collect data from multiple sources to build a comprehensive opportunity index:

Source	Data	Indicator
Census ACS	Demographics	Income, poverty, education
BEA	Regional Accounts	Personal income by state
BLS	Labor Statistics	Unemployment, earnings
FRED	Economic Data	GDP, housing, inflation

[3]:

```python
# =============================================================================
# Fetch Census Demographics for Opportunity Index
# =============================================================================

# Get comprehensive state demographics
demographics_2022 = census.get_demographics_by_state(year=2022)
demographics_2017 = census.get_demographics_by_state(year=2017)

print("  Census ACS Demographics Retrieved:")
print(f"   2022: {len(demographics_2022)} states")
print(f"   2017: {len(demographics_2017)} states")

# Rename columns for clarity
var_names = {
    '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',
}

# Rename and compute derived metrics
for df in [demographics_2017, demographics_2022]:
    for old_name, new_name in var_names.items():
        if old_name in df.columns:
            df.rename(columns={old_name: new_name}, inplace=True)

    # Compute rates
    if 'poverty_pop' in df.columns and 'population' in df.columns:
        df['poverty_rate'] = (df['poverty_pop'] / df['population']) * 100
```

```python
demographics_2022.head()
```

```
{"timestamp": "2025-11-28T09:19:48.837920Z", "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-30", "year": 2022, "variables": 9,
"geography": "state"}
{"timestamp": "2025-11-28T09:19:49.372180Z", "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-30", "year": 2022, "rows": 52}
{"timestamp": "2025-11-28T09:19:49.372805Z", "level": "INFO", "name":
"CensusACSPublicConnector", "message": "Fetching Census ACS data for 2017",
"source": {"file": "census_acs_public.py", "line": 175, "function": "get_data"},
"levelname": "INFO", "taskName": "Task-30", "year": 2017, "variables": 9,
"geography": "state"}
{"timestamp": "2025-11-28T09:19:49.372180Z", "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-30", "year": 2022, "rows": 52}
{"timestamp": "2025-11-28T09:19:49.372805Z", "level": "INFO", "name":
"CensusACSPublicConnector", "message": "Fetching Census ACS data for 2017",
"source": {"file": "census_acs_public.py", "line": 175, "function": "get_data"},
"levelname": "INFO", "taskName": "Task-30", "year": 2017, "variables": 9,
"geography": "state"}
{"timestamp": "2025-11-28T09:19:49.654373Z", "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-30", "year": 2017, "rows": 52}
 Census ACS Demographics Retrieved:
   2022: 52 states
   2017: 52 states
{"timestamp": "2025-11-28T09:19:49.654373Z", "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-30", "year": 2017, "rows": 52}
 Census ACS Demographics Retrieved:
   2022: 52 states
   2017: 52 states
```

[3]:
```
         NAME  population  median_age  white_pop  black_pop  asian_pop  \
0     Alabama     5028092        39.3    3329012    1326341      69808
1      Alaska      734821        35.3     450472      23395      47464
2     Arizona     7172282        38.4    4781702     327077     240642
3    Arkansas     3018669        38.4    2193348     456693      47413
4  California    39356104        37.3   18943660    2202587    5949136
```

```
   hispanic_pop  median_income  poverty_pop state  poverty_rate
0        232407          59609       768897    01     15.292023
1         54890          86370        75227    02     10.237459
2       2297513          72581       916876    04     12.783602
3        243321          56335       475729    05     15.759562
4      15617930          91905      4685272    06     11.904817
```

1.4 3. Economic Indicators from FRED and BLS

Fetch additional economic context to understand mobility barriers.

[4]:
```
# ============================================================================
# Fetch National Economic Indicators from FRED & BLS
# ============================================================================

# GDP growth (proxy for economic opportunity)
gdp = fred.get_series("GDP", start_date="2015-01-01", end_date="2024-12-31")

# Unemployment rate (labor market tightness)
unemployment = bls.get_unemployment_rate()

# CPI (cost of living pressure)
cpi = fred.get_series("CPIAUCSL", start_date="2015-01-01",␣
↪end_date="2024-12-31")

# Average hourly earnings
earnings = bls.get_series("CES0500000003")

print("  Economic Indicators Retrieved:")
print(f"   GDP: {len(gdp)} observations")
print(f"   Unemployment: {len(unemployment)} observations")
print(f"   CPI: {len(cpi)} observations")
print(f"   Earnings: {len(earnings)} observations")

# Compute summary stats
print("\n  National Economic Context:")
print(f"   GDP Growth (2015-2024): {((gdp['value'].iloc[-1] / gdp['value'].
↪iloc[0]) - 1) * 100:.1f}%")
print(f"   CPI Growth: {((cpi['value'].iloc[-1] / cpi['value'].iloc[0]) - 1) *␣
↪100:.1f}%")
print(f"   Current Unemployment: {unemployment['value'].iloc[-1]:.1f}%")
```

```
{"timestamp": "2025-11-28T09:19:49.667927Z", "level": "INFO", "name":
"FREDBasicConnector", "message": "Fetching FRED series: GDP", "source": {"file":
"fred_basic.py", "line": 167, "function": "get_series"}, "levelname": "INFO",
"taskName": "Task-33", "series_id": "GDP", "start_date": "2015-01-01",
"end_date": "2024-12-31"}
{"timestamp": "2025-11-28T09:19:49.881661Z", "level": "INFO", "name":
```

```
"FREDBasicConnector", "message": "Retrieved 40 observations for GDP", "source":
{"file": "fred_basic.py", "line": 197, "function": "get_series"}, "levelname":
"INFO", "taskName": "Task-33", "series_id": "GDP", "rows": 40}
{"timestamp": "2025-11-28T09:19:49.882418Z", "level": "INFO", "name":
"BLSBasicConnector", "message": "Fetching BLS series: LNS14000000", "source":
{"file": "bls_basic.py", "line": 196, "function": "get_series"}, "levelname":
"INFO", "taskName": "Task-33", "series_id": "LNS14000000", "start_year": 2016,
"end_year": 2025}
{"timestamp": "2025-11-28T09:19:49.881661Z", "level": "INFO", "name":
"FREDBasicConnector", "message": "Retrieved 40 observations for GDP", "source":
{"file": "fred_basic.py", "line": 197, "function": "get_series"}, "levelname":
"INFO", "taskName": "Task-33", "series_id": "GDP", "rows": 40}
{"timestamp": "2025-11-28T09:19:49.882418Z", "level": "INFO", "name":
"BLSBasicConnector", "message": "Fetching BLS series: LNS14000000", "source":
{"file": "bls_basic.py", "line": 196, "function": "get_series"}, "levelname":
"INFO", "taskName": "Task-33", "series_id": "LNS14000000", "start_year": 2016,
"end_year": 2025}
{"timestamp": "2025-11-28T09:19:50.073200Z", "level": "INFO", "name":
"BLSBasicConnector", "message": "Retrieved 117 observations for LNS14000000",
"source": {"file": "bls_basic.py", "line": 242, "function": "get_series"},
"levelname": "INFO", "taskName": "Task-33", "series_id": "LNS14000000", "rows":
117}
{"timestamp": "2025-11-28T09:19:50.073795Z", "level": "INFO", "name":
"FREDBasicConnector", "message": "Fetching FRED series: CPIAUCSL", "source":
{"file": "fred_basic.py", "line": 167, "function": "get_series"}, "levelname":
"INFO", "taskName": "Task-33", "series_id": "CPIAUCSL", "start_date":
"2015-01-01", "end_date": "2024-12-31"}
{"timestamp": "2025-11-28T09:19:50.073200Z", "level": "INFO", "name":
"BLSBasicConnector", "message": "Retrieved 117 observations for LNS14000000",
"source": {"file": "bls_basic.py", "line": 242, "function": "get_series"},
"levelname": "INFO", "taskName": "Task-33", "series_id": "LNS14000000", "rows":
117}
{"timestamp": "2025-11-28T09:19:50.073795Z", "level": "INFO", "name":
"FREDBasicConnector", "message": "Fetching FRED series: CPIAUCSL", "source":
{"file": "fred_basic.py", "line": 167, "function": "get_series"}, "levelname":
"INFO", "taskName": "Task-33", "series_id": "CPIAUCSL", "start_date":
"2015-01-01", "end_date": "2024-12-31"}
{"timestamp": "2025-11-28T09:19:50.405684Z", "level": "INFO", "name":
"FREDBasicConnector", "message": "Retrieved 120 observations for CPIAUCSL",
"source": {"file": "fred_basic.py", "line": 197, "function": "get_series"},
"levelname": "INFO", "taskName": "Task-33", "series_id": "CPIAUCSL", "rows":
120}
{"timestamp": "2025-11-28T09:19:50.406425Z", "level": "INFO", "name":
"BLSBasicConnector", "message": "Fetching BLS series: CES0500000003", "source":
{"file": "bls_basic.py", "line": 196, "function": "get_series"}, "levelname":
"INFO", "taskName": "Task-33", "series_id": "CES0500000003", "start_year": 2016,
"end_year": 2025}
{"timestamp": "2025-11-28T09:19:50.405684Z", "level": "INFO", "name":
```

```
"FREDBasicConnector", "message": "Retrieved 120 observations for CPIAUCSL",
"source": {"file": "fred_basic.py", "line": 197, "function": "get_series"},
"levelname": "INFO", "taskName": "Task-33", "series_id": "CPIAUCSL", "rows":
120}
{"timestamp": "2025-11-28T09:19:50.406425Z", "level": "INFO", "name":
"BLSBasicConnector", "message": "Fetching BLS series: CES0500000003", "source":
{"file": "bls_basic.py", "line": 196, "function": "get_series"}, "levelname":
"INFO", "taskName": "Task-33", "series_id": "CES0500000003", "start_year": 2016,
"end_year": 2025}
{"timestamp": "2025-11-28T09:19:50.597616Z", "level": "INFO", "name":
"BLSBasicConnector", "message": "Retrieved 117 observations for CES0500000003",
"source": {"file": "bls_basic.py", "line": 242, "function": "get_series"},
"levelname": "INFO", "taskName": "Task-33", "series_id": "CES0500000003",
"rows": 117}
 Economic Indicators Retrieved:
  GDP: 40 observations
  Unemployment: 117 observations
  CPI: 120 observations
  Earnings: 117 observations

 National Economic Context:
  GDP Growth (2015-2024): 65.1%
  CPI Growth: 35.3%
  Current Unemployment: 4.4%
{"timestamp": "2025-11-28T09:19:50.597616Z", "level": "INFO", "name":
"BLSBasicConnector", "message": "Retrieved 117 observations for CES0500000003",
"source": {"file": "bls_basic.py", "line": 242, "function": "get_series"},
"levelname": "INFO", "taskName": "Task-33", "series_id": "CES0500000003",
"rows": 117}
 Economic Indicators Retrieved:
  GDP: 40 observations
  Unemployment: 117 observations
  CPI: 120 observations
  Earnings: 117 observations

 National Economic Context:
  GDP Growth (2015-2024): 65.1%
  CPI Growth: 35.3%
  Current Unemployment: 4.4%
```

[5]:

```python
# ============================================================================
# Calculate Opportunity Change Index by State
# ============================================================================

# Merge 2017 and 2022 demographics
df_2017 = demographics_2017.set_index('NAME')
df_2022 = demographics_2022.set_index('NAME')
```

```
# Calculate changes
opportunity_df = pd.DataFrame({
    'state': df_2022.index,
    'income_2017': df_2017['median_income'].values,
    'income_2022': df_2022['median_income'].values,
    'poverty_rate_2017': df_2017['poverty_rate'].values,
    'poverty_rate_2022': df_2022['poverty_rate'].values,
    'population_2022': df_2022['population'].values,
})

# Compute change metrics
opportunity_df['income_growth'] = ((opportunity_df['income_2022'] /␣
 ↪opportunity_df['income_2017']) - 1) * 100
opportunity_df['poverty_change'] = opportunity_df['poverty_rate_2022'] -␣
 ↪opportunity_df['poverty_rate_2017']

# Opportunity Score: High income growth + declining poverty = better opportunity
opportunity_df['opportunity_score'] = (
    opportunity_df['income_growth'].rank(pct=True) * 0.5 +
    (-opportunity_df['poverty_change']).rank(pct=True) * 0.5
) * 100

opportunity_df = opportunity_df.sort_values('opportunity_score',␣
 ↪ascending=False)
opportunity_df['rank'] = range(1, len(opportunity_df) + 1)

print("  Opportunity Score Calculated:")
opportunity_df[['state', 'income_growth', 'poverty_change',␣
 ↪'opportunity_score', 'rank']].head(10)
```

```
 Opportunity Score Calculated:
```

[5]:

```
                   state  income_growth  poverty_change  opportunity_score  \
32              New York     311.560051      -31.204451         100.000000
29         New Hampshire     106.180069      -10.147517          96.153846
44                  Utah      86.597185       -9.330509          93.269231
11                Hawaii     116.406090       -8.231683          93.269231
21         Massachusetts      97.833173       -6.530596          89.423077
7               Delaware      69.795368       -9.311553          89.423077
23             Minnesota      73.098875       -7.189227          88.461538
45               Vermont      58.454292       -8.977004          83.653846
8   District of Columbia     118.888793       -2.896905          80.769231
1                 Alaska      67.572077       -3.945857          78.846154

    rank
32     1
```

```
29      2
44      3
11      4
21      5
7       6
23      7
45      8
8       9
1      10
```

1.5 4. Regional Economic Analysis with KRL Models

Using **krl_models** for regional economic analysis - specifically the Location Quotient and Shift-Share methods.

[6]:
```
# ============================================================================
# Demonstrate KRL Model Zoo: Location Quotient
# ============================================================================

# Create synthetic regional employment data for demonstration
# In production, this would come from BLS QCEW or CBP data

np.random.seed(42)

# Create sample employment data by region and industry
regions = opportunity_df['state'].head(10).tolist()
industries = ['Manufacturing', 'Services', 'Technology', 'Healthcare', 'Retail']

employment_data = []
for region in regions:
    region_factor = np.random.uniform(0.8, 1.2)
    for industry in industries:
        industry_factor = {'Manufacturing': 0.8, 'Services': 1.0, 'Technology':␣
↪1.2,
                          'Healthcare': 1.1, 'Retail': 0.9}[industry]
        employment = int(10000 * region_factor * industry_factor * np.random.
↪uniform(0.7, 1.3))
        employment_data.append({
            'region': region,
            'industry': industry,
            'employment': employment
        })

employment_df = pd.DataFrame(employment_data)

print("  Sample Regional Employment Data:")
print(f"   Regions: {len(regions)}")
print(f"   Industries: {len(industries)}")
employment_df.head(10)
```

```
 Sample Regional Employment Data:
   Regions: 10
   Industries: 5
```

```
[6]:           region       industry  employment
    0        New York  Manufacturing        9653
    1        New York       Services       10820
    2        New York     Technology       12072
    3        New York     Healthcare        8291
    4        New York         Retail        6783
    5   New Hampshire  Manufacturing        8032
    6   New Hampshire       Services        8731
    7   New Hampshire     Technology       11112
    8   New Hampshire     Healthcare        6450
    9   New Hampshire         Retail        9498
```

```
[7]: # ============================================================================
     # Apply Location Quotient Model
     # ============================================================================

     # Initialize and fit Location Quotient Model
     lq_model = LocationQuotientModel(
         industry_col='industry',
         employment_col='employment',
         region_col='region'
     )

     # Fit the model
     lq_model.fit(employment_df)

     # Get Location Quotient matrix
     lq_matrix = lq_model.get_location_quotients()

     print("  Location Quotient Matrix (LQ > 1 = Regional Specialization):")
     print(lq_matrix.round(2))

     # Identify specialized industries for top region
     top_region = regions[0]
     specialized = lq_model.get_specialized_industries(top_region, threshold=1.1)
     print(f"\n  {top_region} Specializations (LQ > 1.1):")
     for industry, lq in specialized.items():
         print(f"    • {industry}: LQ = {lq:.2f}")
```

```
 Location Quotient Matrix (LQ > 1 = Regional Specialization):
industry              Healthcare  Manufacturing  Retail  Services  Technology
```

```
region
Alaska                    0.88          1.40   1.01     0.89        0.92
Delaware                  1.25          0.72   1.08     0.70        1.15
District of Columbia      1.14          0.67   1.04     1.12        0.97
Hawaii                    1.02          0.94   1.00     1.21        0.85
Massachusetts             1.16          1.18   0.74     0.88        1.03
Minnesota                 0.88          0.79   1.05     1.23        1.02
New Hampshire             0.70          1.11   1.18     1.02        1.03
New York                  0.83          1.23   0.77     1.16        1.03
Utah                      1.07          0.93   1.15     0.96        0.91
Vermont                   0.92          1.19   0.98     0.87        1.06

 New York Specializations (LQ > 1.1):
   • Manufacturing: LQ = 1.23
   • Services: LQ = 1.16
```

1.6 5. Identifying Mobility Deserts

States with low opportunity scores and declining economic conditions are classified as "mobility deserts."

[8]:
```
# ==============================================================================
# Classify States as Opportunity Zones or Mobility Deserts
# ==============================================================================

def classify_opportunity(score):
    if score >= 75:
        return 'Opportunity Zone'
    elif score >= 50:
        return 'Stable'
    elif score >= 25:
        return 'At Risk'
    else:
        return 'Mobility Desert'

opportunity_df['classification'] = opportunity_df['opportunity_score'].
 ↪apply(classify_opportunity)

# Count by classification
class_counts = opportunity_df['classification'].value_counts()

print("  State Classification by Opportunity Level:")
for classification, count in class_counts.items():
    pct = count / len(opportunity_df) * 100
    print(f"   • {classification}: {count} states ({pct:.0f}%)")

# Show mobility deserts
deserts = opportunity_df[opportunity_df['classification'] == 'Mobility Desert']
```

```
print(f"\n  Mobility Deserts ({len(deserts)} states):")
for _, row in deserts.iterrows():
    print(f"   • {row['state']}: Income growth {row['income_growth']:.1f}%,␣
↪Poverty change {row['poverty_change']:+.1f}pts")
```

```
State Classification by Opportunity Level:
  • At Risk: 18 states (35%)
  • Opportunity Zone: 13 states (25%)
  • Stable: 12 states (23%)
  • Mobility Desert: 9 states (17%)

Mobility Deserts (9 states):
  • Montana: Income growth 1.6%, Poverty change +1.3pts
  • Louisiana: Income growth 3.1%, Poverty change +3.6pts
  • Mississippi: Income growth -7.1%, Poverty change +2.9pts
  • Arkansas: Income growth -0.6%, Poverty change +4.1pts
  • Florida: Income growth -10.8%, Poverty change +2.7pts
  • Ohio: Income growth -9.2%, Poverty change +3.2pts
  • New Mexico: Income growth -3.6%, Poverty change +7.1pts
  • West Virginia: Income growth -25.6%, Poverty change +5.6pts
  • Puerto Rico: Income growth -63.3%, Poverty change +30.6pts
```

[9]:
```
# ============================================================================
# Visualization: Opportunity Score 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'
}

opportunity_df['state_abbrev'] = opportunity_df['state'].map(state_abbrev)

fig = px.choropleth(
    opportunity_df,
    locations='state_abbrev',
    locationmode='USA-states',
    color='opportunity_score',
    scope='usa',
    color_continuous_scale='RdYlGn',
    hover_name='state',
    hover_data={'income_growth': ':.1f', 'poverty_change': ':.1f',␣
↪'opportunity_score': ':.0f'},
    title='Economic Opportunity Score by State (2017-2022)',
)

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

1.7 6. Causal Analysis with KRL Policy Toolkit

Using **krl_policy** for causal inference to understand policy impacts on mobility.

1.7.1 Performance Optimization Note

The `TreatmentEffectEstimator` has been optimized with parallel processing for maximum precision:

Feature	Benefit	Configuration
High Bootstrap Count	Stable CI endpoints, minimal p-value jitter	`n_bootstrap=2000`
Parallel Processing	Near-instant computation across all cores	`n_jobs=-1`
Progress Tracking	Real-time iteration monitoring	Automatic via tqdm

Bootstrap Accuracy Hierarchy: - **B = 200-400**: Minimal viability (tails wobble, CI jitter) - **B = 800-1200**: Stability zone (endpoints stop drifting) - **B = 2000**: **High-fidelity inference** (stable p-values, tight CIs) ← **This notebook** - **B > 5000**: Asymptotic luxury (diminishing returns)

Why B=2000? Bootstrap error declines at $O(1/\sqrt{B})$. With parallel processing removing runtime penalty, B=2000 delivers maximum practical precision before diminishing returns dominate.

See TREATMENT_EFFECT_OPTIMIZATION.md for technical details.

[10]:
```
# ============================================================================
# Demonstrate KRL Policy: Treatment Effect Estimation
# ============================================================================

# Create sample data for policy evaluation
# Simulate: States that implemented workforce programs vs. control states

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

# Assign treatment (workforce program) to states with lower opportunity
opportunity_df['treatment'] = (opportunity_df['opportunity_score'] < 50).
 ↪astype(int)

# Outcome: Simulated employment growth
opportunity_df['outcome'] = (
    opportunity_df['income_growth'] * 0.3 +
    opportunity_df['treatment'] * 5 +  # Treatment effect
    np.random.normal(0, 3, n_states)
)

print("  Policy Evaluation Dataset:")
print(f"   Treatment group (workforce program): {opportunity_df['treatment'].
 ↪sum()} states")
print(f"   Control group: {(1 - opportunity_df['treatment']).sum()} states")
print(f"\n   Mean outcome (treated):␣
 ↪{opportunity_df[opportunity_df['treatment']==1]['outcome'].mean():.2f}")
print(f"   Mean outcome (control):␣
 ↪{opportunity_df[opportunity_df['treatment']==0]['outcome'].mean():.2f}")
```

```
 Policy Evaluation Dataset:
   Treatment group (workforce program): 27 states
   Control group: 25 states

   Mean outcome (treated): 5.72
   Mean outcome (control): 20.42
```

[11]:
```
# ============================================================================
# Apply Treatment Effect Estimator - HIGH-PRECISION VERSION
# ============================================================================

print("  High-Precision Configuration:")
print("    • Bootstrap Iterations: 2000 (maximum stability)")
print("    • Parallel Processing: All CPU cores")
print("    • Adaptive Bootstrap: Disabled (using full iteration count)")
print("    • Progress Tracking: Real-time via tqdm\n")
```

```
# Prepare data for treatment effect estimation
te_data = opportunity_df[['income_2017', 'poverty_rate_2017', 'treatment',␣
 ↪'outcome']].copy()
te_data = te_data.fillna(0)

# Initialize Treatment Effect Estimator for maximum precision
te_estimator = TreatmentEffectEstimator(
    method='doubly_robust',
    treatment_col='treatment',
    outcome_col='outcome',
    n_bootstrap=2000,          # High iteration count for stable CI endpoints
    adaptive_bootstrap=False, # Use full 2000 iterations
    n_jobs=-1,                 # Parallel across all CPU cores
    random_state=42
)

print("  Fitting doubly robust estimator with 2000 bootstrap iterations...")
print("   (Parallel processing makes this fast despite high iteration count)\n")

import time
start_time = time.time()

# Estimate Average Treatment Effect
te_estimator.fit(
    data=te_data,
    treatment_col='treatment',
    outcome_col='outcome',
    covariate_cols=['income_2017', 'poverty_rate_2017']
)

elapsed = time.time() - start_time

print(f"\n  Estimation complete in {elapsed:.2f} seconds")
print("\n  Treatment Effect Estimation (Doubly Robust):")
print(f"   Average Treatment Effect (ATE): {te_estimator.effect_:.2f}")
print(f"   Standard Error: {te_estimator.std_error_:.2f}")
print(f"   95% CI: [{te_estimator.ci_[0]:.2f}, {te_estimator.ci_[1]:.2f}]")
print(f"   P-value: {te_estimator.p_value_:.4f}")

if te_estimator.p_value_ < 0.05:
    print("\n     Statistically significant treatment effect detected")
else:
    print("\n     Treatment effect not statistically significant")

print("\n  Why B=2000?")
print("   • Bootstrap error declines at O(1/√B)")
```

```python
print("   • B=2000 ensures stable CI endpoints and p-values")
print("   • Parallel processing removes runtime penalty")
print("   • Beyond 2000, diminishing returns dominate")
```

```
 High-Precision Configuration:
   • Bootstrap Iterations: 2000 (maximum stability)
   • Parallel Processing: All CPU cores
   • Adaptive Bootstrap: Disabled (using full iteration count)
   • Progress Tracking: Real-time via tqdm

 Fitting doubly robust estimator with 2000 bootstrap iterations…
   (Parallel processing makes this fast despite high iteration count)

{"timestamp": "2025-11-28T09:19:51.527937Z", "level": "INFO", "name":
"krl_policy.estimators.treatment_effect", "message": "Fitted doubly_robust:
ATE=-4.2258 (SE=2.7366, p=0.1226)", "source": {"file": "treatment_effect.py",
"line": 284, "function": "fit"}, "levelname": "INFO", "taskName": "Task-54"}

 Estimation complete in 0.01 seconds

 Treatment Effect Estimation (Doubly Robust):
   Average Treatment Effect (ATE): -4.23
   Standard Error: 2.74
   95% CI: [-9.59, 1.14]
   P-value: 0.1226

     Treatment effect not statistically significant

 Why B=2000?
   • Bootstrap error declines at O(1/√B)
   • B=2000 ensures stable CI endpoints and p-values
   • Parallel processing removes runtime penalty
   • Beyond 2000, diminishing returns dominate
```

1.8 7. Visualization Dashboard

[12]:
```python
# ============================================================================
# Dashboard: Economic Mobility Indicators
# ============================================================================

fig = make_subplots(
    rows=2, cols=2,
    subplot_titles=(
        'Opportunity Score Distribution',
        'Income Growth vs Poverty Change',
        'Classification by State Count',
        'Top/Bottom States by Opportunity'
    ),
    specs=[[{"type": "histogram"}, {"type": "scatter"}],
           [{"type": "pie"}, {"type": "bar"}]]
)

# Histogram of opportunity scores
fig.add_trace(
    go.Histogram(x=opportunity_df['opportunity_score'], nbinsx=20,␣
 ↪marker_color='#0077BB'),
    row=1, col=1
)

# Scatter: Income growth vs poverty change
fig.add_trace(
    go.Scatter(
        x=opportunity_df['income_growth'],
        y=opportunity_df['poverty_change'],
        mode='markers',
        marker=dict(color=opportunity_df['opportunity_score'],␣
 ↪colorscale='RdYlGn', size=10),
        text=opportunity_df['state'],
        hovertemplate='%{text}<br>Income: %{x:.1f}%<br>Poverty: %{y:.1f}pts'
    ),
    row=1, col=2
)

# Pie: Classification counts
class_counts = opportunity_df['classification'].value_counts()
fig.add_trace(
    go.Pie(labels=class_counts.index, values=class_counts.values,
           marker=dict(colors=['#009988', '#EE7733', '#CC3311', '#0077BB'])),
    row=2, col=1
)

# Bar: Top/bottom states
top_5 = opportunity_df.head(5)
bottom_5 = opportunity_df.tail(5)
combined = pd.concat([top_5, bottom_5])

fig.add_trace(
    go.Bar(x=combined['state'], y=combined['opportunity_score'],
           marker_color=['#009988']*5 + ['#CC3311']*5),
    row=2, col=2
)

fig.update_layout(height=700, showlegend=False, title_text='Economic Mobility␣
 ↪Analysis Dashboard')
fig.show()
```

1.9 8. Key Insights & Policy Implications

[13]:
```
# ============================================================================
# Key Insights Summary
# ============================================================================

print("=" * 65)
print("  KEY INSIGHTS: Economic Mobility Deserts Analysis")
print("=" * 65)

print(f"\n  Geographic Scope: 50 States + DC")
print(f"  Analysis Period: 2017-2022")

print(f"\n  NATIONAL TRENDS:")
print(f"    • Avg Income Growth: {opportunity_df['income_growth'].mean():.1f}%")
print(f"    • Avg Poverty Change: {opportunity_df['poverty_change'].mean():+.2f}␣
  ↪pts")
print(f"    • States Improving: {(opportunity_df['opportunity_score'] >= 50).
  ↪sum()}")
print(f"    • States Declining: {(opportunity_df['opportunity_score'] < 50).
  ↪sum()}")

print(f"\n  TOP 5 OPPORTUNITY STATES:")
for _, row in opportunity_df.head(5).iterrows():
    print(f"    {row['rank']}. {row['state']}: Score {row['opportunity_score']:.
  ↪0f}")

print(f"\n  MOBILITY DESERTS:")
for _, row in opportunity_df.tail(5).iterrows():
    print(f"    {row['rank']}. {row['state']}: Score {row['opportunity_score']:.
  ↪0f}")

print("\n" + "=" * 65)
print("  POLICY RECOMMENDATIONS")
print("=" * 65)
print("""
1. TARGETED INTERVENTIONS: Focus workforce programs on
   mobility desert states showing declining opportunity.

2. EARLY WARNING: Use Location Quotient analysis to
   identify regions losing key industries.

3. CAUSAL EVALUATION: Apply Difference-in-Differences
   to measure policy effectiveness over time.

4. GRANULAR ANALYSIS: Upgrade to Professional tier for
   county/tract-level mobility desert identification.
""")
```

```
=================================================================
 KEY INSIGHTS: Economic Mobility Deserts Analysis
=================================================================

 Geographic Scope: 50 States + DC
 Analysis Period: 2017-2022

 NATIONAL TRENDS:
   • Avg Income Growth: 36.1%
   • Avg Poverty Change: -1.69 pts
   • States Improving: 25
   • States Declining: 27

 TOP 5 OPPORTUNITY STATES:
   1. New York: Score 100
   2. New Hampshire: Score 96
   3. Utah: Score 93
   4. Hawaii: Score 93
   5. Massachusetts: Score 89

 MOBILITY DESERTS:
   48. Florida: Score 11
   49. Ohio: Score 10
   50. New Mexico: Score 9
   51. West Virginia: Score 5
   52. Puerto Rico: Score 2

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

1. TARGETED INTERVENTIONS: Focus workforce programs on
   mobility desert states showing declining opportunity.

2. EARLY WARNING: Use Location Quotient analysis to
   identify regions losing key industries.

3. CAUSAL EVALUATION: Apply Difference-in-Differences
   to measure policy effectiveness over time.

4. GRANULAR ANALYSIS: Upgrade to Professional tier for
   county/tract-level mobility desert identification.
```

1.10 9. Data Provenance & Next Steps

1.10.1 Data Sources Used

Source	Package	Data
Census ACS	`CensusACSPublicConnector`	State demographics
FRED	`FREDBasicConnector`	GDP, CPI
BLS	`BLSBasicConnector`	Unemployment, earnings

1.10.2 KRL Suite Components Demonstrated

Package	Component	Use Case
`krl_models`	`LocationQuotientModel`	Regional industry specialization
`krl_policy`	`TreatmentEffectEstimator`	Policy impact evaluation

1.10.3 Next Steps

- **04-environmental-justice-health.ipynb**: Environmental burden analysis
- **10-urban-resilience-dashboard.ipynb**: Complete integration workflow

1.10.4 Professional Tier Features

```
from krl_data_connectors.professional import LODESConnector
from krl_policy import SyntheticControlMethod

# Tract-level job access data
lodes = LODESConnector(license_key="YOUR_KEY")
job_access = lodes.get_rac(state="CA", year=2021)

# Synthetic control for policy evaluation
scm = SyntheticControlMethod()
scm.fit(panel_data, treated_unit="California", treatment_time=2018)
```

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