

PSEO
COALITION

Advanced PSEO Data Use

APIs and Data Integration for Deeper Analysis

August 28, 2025

Presenter and Host



Christopher Peña, PhD

Assistant Professor of the Practice, University of Denver
Intern Supervisor, PSEO Coalition



Annika Many

President & CEO, EDU-PM
Member Engagement Consultant, PSEO Coalition

Introduction

PSEO Coalition Summer Webinar Series

June 26

**Getting Started with the
PSEO Explorer:
Navigating
Postsecondary
Employment Outcomes**

July 29

**Using PSEO Data in
Practice:
Tools, Insights, and Key
Considerations**

August 28

**Advanced PSEO Data
Use:
APIs and Data
Integration for Deeper
Analysis**



Agenda

- Need for Data Pipelines
- Supplemental Data for Context
 - IPEDS (Data Download)
 - Living Wage Calculator (Web Scraping)
 - U.S. Census Bureau (API)
- PSEO Data
 - Working with the PSEO API
 - Python Demo



Learning Objectives

By the end of this webinar, participants will be able to:

- Identify and access key data sources relevant to postsecondary outcomes.
- Explain the role of APIs in accessing structured data and recognize how they differ from bulk downloads or web tools.
- Retrieve, structure, and integrate postsecondary outcomes data for analysis.
- Apply a reproducible Python workflow to query and merge postsecondary outcomes data for policy, planning, or institutional decision-making.



Need for Data Pipelines

What is a Data Pipeline?

A **data pipeline** is an automated series of steps designed to move, transform, and process raw data from various sources into a usable format at a destination.

EXAMPLE

An IR analyst wants to create a single dataset for peer analysis with key institutional indicators such as retention rate, graduation rate, and spend per student FTE using data from IPEDS surveys.



Planning the Task Through Pseudocode

Pseudocode is a simplified method of describing the steps of an algorithm using plain language mixed with language-independent programming conventions.

WHY WRITE PSEUDOCODE?

- Clarify logic
- Plan before coding
- Communication
- Debugging



IPEDS

What is IPEDS?

The Integrated Postsecondary Education Data System (IPEDS) is a set of institutional data collected through surveys each year for colleges and universities that receive Title IV federal financial aid.

ACCESSING THE DATA

- Web analysis tools
- Custom data files
- Complete data files
- Access database



Locating Directory Information

IPEDS Integrated Postsecondary Education Data System Data Tools | Help Desk 1 866-558-0658

[Start over](#) [Save session](#) [Help](#) [MAIN MENU](#)

Complete Data Files Data Release Info

Years & Surveys

2023 Institutional Characteristics [Continue](#)

Data files are available in ZIP format.

Year	Survey	Title	Data File	Stata Data File	Programs	Dictionary
2023	Institutional Characteristics	Directory information (updated January 2025)	HD2023	HD2023_STATA	SPSS, SAS, STATA	Dictionary
2023	Institutional Characteristics	Educational offerings, organization, services and athletic associations	IC2023	IC2023_STATA	SPSS, SAS, STATA	Dictionary
2023	Institutional Characteristics	Student charges for academic year programs	IC2023_AY	IC2023_AY_STATA	SPSS, SAS, STATA	Dictionary
2023	Institutional Characteristics	Student charges by program (vocational programs)	IC2023_PY	IC2023_PY_STATA	SPSS, SAS, STATA	Dictionary



Breaking it Down

1. Identify the URL to the requested Data Center file.
2. Download the ZIP file from the URL.
3. Unzip the ZIP file.
4. Extract the CSV file in the extracted folder.
5. Delete the ZIP file.
6. Read the CSV file into a Pandas data frame.

PRO CHALLENGE

Build a web scraper to create a dictionary of file URLs.



Sample Code

Step 3: Fetch the most recent data from the IPEDS Institutional Characteristics survey.

```
# Define the parameters for downloading and unzipping the IPEDS survey file.
url = 'https://nces.ed.gov/ipeds/datacenter/data/HD2023.zip'
zip_filename = 'HD2023.zip'

# Download the ZIP file.
print(f"Downloading {zip_filename}...")
response = requests.get(url)
response.raise_for_status()

# Save the ZIP file to disk.
with open(zip_filename, "wb") as f:
    f.write(response.content)
print(f"Downloaded {zip_filename}.")

# Extract the contents of the ZIP file.
print(f"Extracting {zip_filename}...")
with zipfile.ZipFile(zip_filename, 'r') as zip_ref:
    zip_ref.extractall()
print(f"Extracted contents of {zip_filename}.")

# Delete the ZIP file.
os.remove(zip_filename)
print(f"Deleted {zip_filename}.")

# Read the data into a data frame.
ipeds = pd.read_csv("HD2023.csv", encoding="latin1", dtype={"UNITID":str, "OPEID":str})
```



Joining the Data

To join the PSEO data to IPEDS data, we need a join field: **OPEID**

METHOD

1. Merge the PSEO data to the IPEDS IC survey data on *institution* = *OPEID*.
2. To merge **other** IPEDS survey data, merge those data to the IPEDS IC survey data first on *unitid*, then merge to the PSEO data.



Living Wage Calculator

What is the Living Wage Calculator?

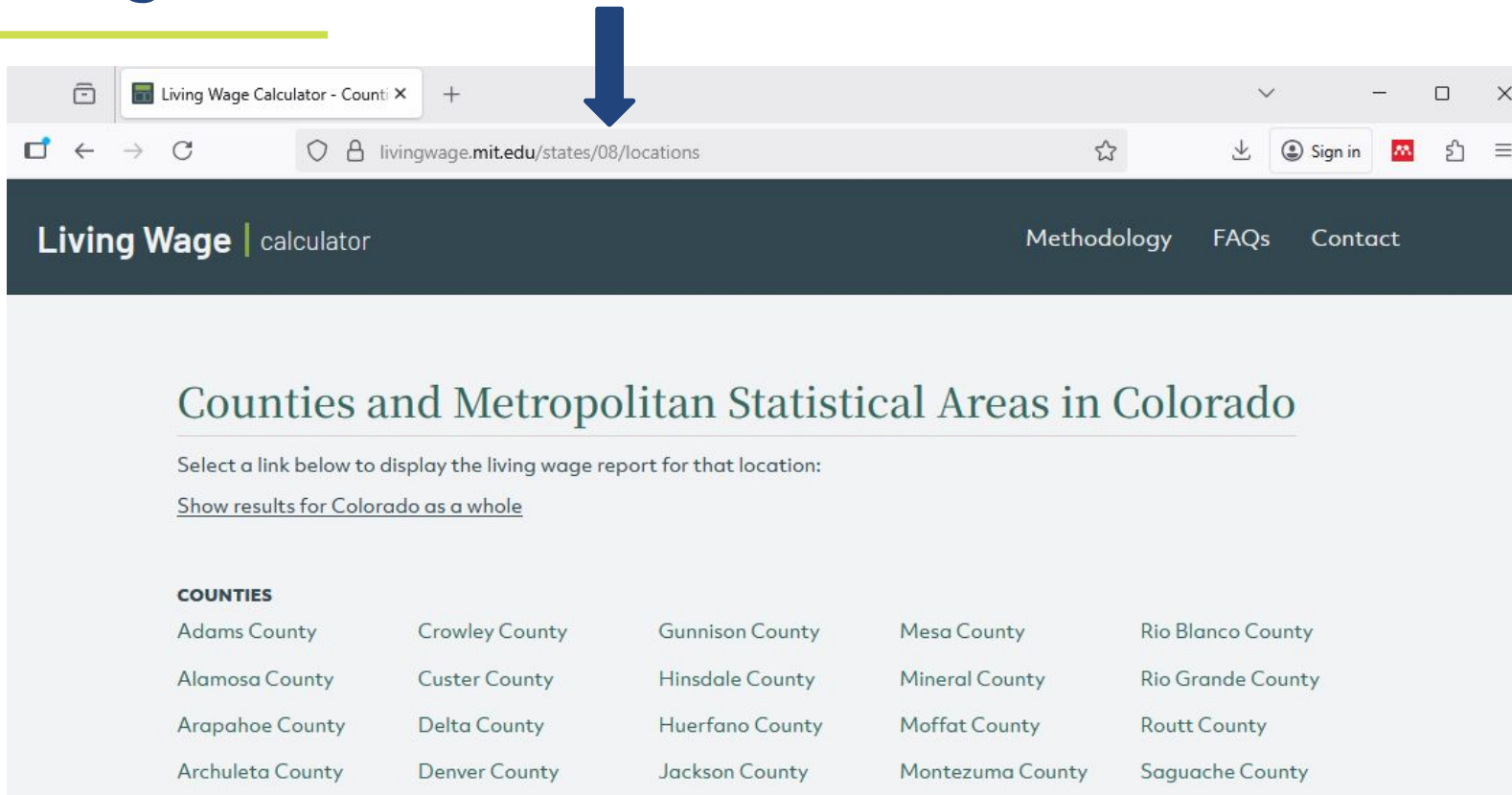
The MIT Living Wage Calculator estimates the local wage that a full-time worker requires to cover the costs of their family's basic needs where they live.

LEVELS OF ANALYSIS

- State
- County
- Metropolitan Statistical Area (MSA)



Building a URL



A screenshot of a web browser window. The address bar shows the URL `livingwage.mit.edu/states/08/locations`. A large blue arrow points down to the address bar. The browser tab is titled "Living Wage Calculator - Counti". The page header includes "Living Wage | calculator" and navigation links for "Methodology", "FAQs", and "Contact". The main heading is "Counties and Metropolitan Statistical Areas in Colorado". Below this, it says "Select a link below to display the living wage report for that location:" and provides a link "[Show results for Colorado as a whole](#)". A section titled "COUNTIES" lists various counties in a grid.

Adams County	Crowley County	Gunnison County	Mesa County	Rio Blanco County
Alamosa County	Custer County	Hinsdale County	Mineral County	Rio Grande County
Arapahoe County	Delta County	Huerfano County	Moffat County	Routt County
Archuleta County	Denver County	Jackson County	Montezuma County	Saguache County



Locating the Data



	1 ADULT				2 ADULTS (1 WORKING)				2 ADULTS (BOTH WORKING)		
	0 Children	1 Child	2 Children	3 Children	0 Children	1 Child	2 Children	3 Children	0 Children	1 Child	2 Children
Living Wage	\$27.01	\$49.85	\$64.45	\$81.99	\$36.66	\$43.11	\$47.34	\$55.43	\$18.33	\$27.28	\$34.64
Poverty Wage	\$7.52	\$10.17	\$12.81	\$15.46	\$10.17	\$12.81	\$15.46	\$18.10	\$5.08	\$6.41	\$7.73
Minimum Wage	\$14.81	\$14.81	\$14.81	\$14.81	\$14.81	\$14.81	\$14.81	\$14.81	\$14.81	\$14.81	\$14.81



Breaking It Down

1. Declare the base URL.
2. Create a dictionary of state codes and FIPS codes.
3. For each state in the dictionary, build a URL.
4. For each URL from the dictionary, go to the state webpage.
5. For each state page, read in the list of MSAs into a dictionary.
6. For each MSA in the dictionary, build a URL.
7. For each URL from the dictionary, go to the MSA webpage.
8. Locate the living wage calculation table.
9. Read the defined columns from the table into a Pandas data frame.



Pseudocode

1. Import required libraries.
2. Define FIPS codes and state names.
3. Define base URL for retrieving the data.
4. Set headers to simulate a browser request.
5. Define function to parse currency values.
6. Define function to retrieve living wage data.
7. Define expected wage keys.
8. Initialize data storage.
9. Loop through states to retrieve living wage data.
10. Write data to a CSV file.



Sample Code

Step 6: Define function to retrieve living wage data.

```
def get_all_annual_wages(url):
    try:
        response = requests.get(url, headers=headers)
        response.raise_for_status()
        soup = BeautifulSoup(response.text, 'html.parser')
        tables = soup.find_all('table')

        if not tables:
            print(f"No tables found at {url}")
            return {}

        # We know the data is in the third row (index 2)
        rows = tables[0].find_all('tr')
        if len(rows) < 3:
            print(f"Expected row not found in table at {url}")
            return {}

        wage_row = rows[2] # Third row
        cols = wage_row.find_all(['td', 'th'])

        # Expected 12 columns of wage data starting from index 1
        if len(cols) < 13:
            print(f"Not enough columns in wage row at {url}")
            return {}

        # Define keys in the exact order they appear in the table
        keys = [
            "1adult_0children", "1adult_1child", "1adult_2children", "1adult_3children",
            "2adults1working_0children", "2adults1working_1child", "2adults1working_2children", "2adults1working_3children",
            "2adults2working_0children", "2adults2working_1child", "2adults2working_2children", "2adults2working_3children"
        ]

        wages = {}
        for i in range(12):
            value = parse_currency(cols[i + 1].get_text(strip=True)) # +1 to skip label column
            wages[keys[i]] = round(value * 2080, 2) if value is not None else "N/A"

        return wages
```



Joining the Data

To join the PSEO data to the living wage data, we need a join field: **State**

METHOD

1. Merge PSEO data with `label_institution.csv` on *institution* to get *institution_state* (state code) and *FIPS*.
2. Merge PSEO data to living wage data on *institution_state* or *FIPS*, depending on what is available.



U.S. Census Bureau

U.S. Census Bureau Data

The U.S. Census Bureau provides a wide variety of data from sources such as the American Community Survey (ACS).

ACCESSING THE DATA

- Table builder
- IPUMS
- API



What is an API?

An Application Programming Interface (API) is a set of rules and tools that allow different pieces of software to communicate with each other.

APIs often return data in JSON format.

BASIC COMPONENTS

- Request – How to ask for the data. (GET/POST)
- Response – What you'll get back.



API As a URL

https://api.census.gov/data/timeseries/pseo/earnings?get=Y1_P50_EARNINGS,LABEL_INSTITUTION&for=us:1

JSON		Raw Data	Headers
Save		Copy	Collapse All Expand All Filter JSON
▼ 0:			
0:	"Y1_P50_EARNINGS"		
1:	"LABEL_INSTITUTION"		
2:	"us"		
▼ 1:			
0:	"44129"		
1:	"Institutions in Alabama"		
2:	"1"		
▼ 2:			
0:	"43247"		
1:	"Institutions in Arizona"		
2:	"1"		
▼ 3:			
0:	"40501"		
1:	"Institutions in Colorado"		
2:	"1"		




When to Use an API

- Fresh, real-time data
- Automation and efficiency
- Selective access (only what you need)
- Consistency and standardization
- Scalability
- Access to dynamic services (not just data)



Before You Begin

 An official website of the United States government [Here's how you know](#) ▾



Request a U.S. Census Data API Key

☐ I agree to the [terms of service](#)

REQUEST KEY



Pseudocode

1. Import required libraries.
2. Define the API key and survey year.
3. Define the base URL for retrieving the data.
4. Define the parameters for retrieving the data.
5. Use the API to call for the data.
6. Write the data to a CSV file.



Sample Code

Step 3: Define the survey year for retrieving the data.

```
year = "2023"
```

Step 4: Define the base URL for retrieving the data.

```
base_url = f"https://api.census.gov/data/{year}/acs/acs1/subject"
```

Step 5: Define the parameters for retrieving the data.

```
params = {  
    "get": "NAME,S1501_C01_061E", # Table and column reference for the data.  
    "for": "state:*",  
    "key": api_key  
}
```



Sample Code

Step 6: Use the API to call for the data.

```
response = requests.get(base_url, params=params)

if response.status_code == 200:
    data = response.json()
    headers = data[0]
    rows = data[1:]
    df = pd.DataFrame(rows, columns=headers)
    df = df.rename(columns={
        "state": "State Code",
        "NAME": "State Name",
        "S1501_C01_061E": "Median HS Grad Earnings",
    })
    df["Median HS Grad Earnings"] = pd.to_numeric(df["Median HS Grad Earnings"], errors='coerce')
    df = df.sort_values(by="State Code", ascending=True)

    column_order = ["State Code", "State Name", "Median HS Grad Earnings"]

    df = df[column_order]

    df.to_csv("median_hs_earnings_by_state.csv", index=False)
    print("Data saved to median_high_school_earnings_by_state.csv")
else:
    print(f"Error fetching data: {response.status_code} - {response.text}")
```



Joining the Data

To merge the PSEO data to the census earnings data, we need a join field: **State**

METHOD

1. Merge PSEO data with `label_institution.csv` on *institution* to get *institution_state* (state code) and *FIPS*.
2. Merge PSEO data to census data on *institution_state* or *FIPS*, depending on what is available.



PSEO API

PSEO Earnings – Selected Variables

Name	Label
AGG_LEVEL_PSEO	Aggregation Level
CIP_LEVEL	CIP Code Level
CIPCODE	CIP Code
DEGREE_LEVEL	Degree Level
for	Census API FIPS 'for' clause
GEO_ID	Geographic Identifier Code
GEOCOMP	GEO_ID Component
GRAD_COHORT	First Year of Graduation Cohort
GRAD_COHORT_YEARS	Number of Graduation Cohort Years
in	Census API FIPS 'in' clause
INST_LEVEL	Institution Level
INST_STATE	FIPS Code of State of the Institution
INSTITUTION	Institution
NATION	Nation
SUMLEVEL	Summary Level code
ucgid	Uniform Census Geography Identifier clause
Y1_GRADS_EARN	Count of Employed Graduates in Year 1
Y1_IPEDS_COUNT	Count of IPEDS Reported Graduates of Programs Included in Year 1 Earnings
Y1_P25_EARNINGS	Earnings 25th Percentile in Year 1 (2022 Dollars)
Y1_P50_EARNINGS	Earnings 50th Percentile in Year 1 (2022 Dollars)
Y1_P75_EARNINGS	Earnings 75th Percentile in Year 1 (2022 Dollars)



PSEO Flows – Selected Variables

Name	Label
<u>AGG LEVEL PSEO</u>	Aggregation Level
<u>CIP LEVEL</u>	CIP Code Level
<u>CIPCODE</u>	CIP Code
<u>DEGREE LEVEL</u>	Degree Level
<u>DIVISION</u>	Division
<u>for</u>	Census API FIPS 'for' clause
<u>GEO ID</u>	Geographic Identifier Code
<u>GEOCOMP</u>	GEO_ID Component
<u>GRAD COHORT</u>	First Year of Graduation Cohort
<u>GRAD COHORT YEARS</u>	Number of Graduation Cohort Years
<u>in</u>	Census API FIPS 'in' clause
<u>INST LEVEL</u>	Institution Level
<u>INST STATE</u>	FIPS Code of State of the Institution
<u>INSTITUTION</u>	Institution
<u>NAICS</u>	NAICS Industry Code
<u>NATION</u>	Nation
<u>SUMLEVEL</u>	Summary Level code
<u>ucgid</u>	Uniform Census Geography Identifier clause
<u>Y1 GRADS EMP</u>	Count of Employed Graduates in Year 1
<u>Y1 GRADS EMP INSTATE</u>	Count of Graduates Employed in Same State as Educational Institution in Year 1



Sample Code

3. Get directory data for Arapahoe Community College.

```
# Create a list to store the fields to retrieve.
fields = [
    "INSTITUTION",
    "LABEL_INSTITUTION",
    "INST_STATE",
    "LABEL_INST_STATE"
]

# List the parameters for the GET request.
params = {
    "get": ", ".join(fields),      # Concatenate the list of fields into a single string.
    "INSTITUTION": "00134600",    # Filter records to Arapahoe Community College only.
    "for": "us:1",                # Required geography for the API.
    "key": key
}

# Send the GET request.
response = requests.get(url, params=params)

# Check the response for errors. If none, write the data to a CSV file and display the first few records.
if response.status_code == 200:
    data = response.json()
    # Convert to pandas DataFrame
    df = pd.DataFrame(data[1:], columns=data[0])
    df.to_csv("pseoe_institutions_acc.csv", index=False)
    print(df)
else:
    print(f"Error {response.status_code}: {response.text}")
```



Use the Metadata

Post-Secondary Employment Outcomes (PSEO)

Post-Secondary Employment Outcomes (PSEO) are experimental tabulations developed by researchers at the U.S. Census Bureau. PSEO data provide earnings and employment outcomes for college and university graduates by degree level, degree major, post-secondary institution, and state of institution. These statistics are generated by matching university transcript data with a national database of jobs, using state-of-the-art confidentiality protection mechanisms to protect the underlying data.

The PSEO are made possible through data sharing partnerships between universities, university systems, State Departments of Education, State Labor Market Information offices, and the U.S. Census Bureau. PSEO data are available for post-secondary institutions whose transcript data have been made available to the Census Bureau through a data-sharing agreement.

Download Public-Use Data

We release two classes of files for each of the tabulations, Graduate Earnings and Employment Flows:

- Comprehensive dataset, which includes all institutions and crossings
- State datasets, which include all institutions in a state and are a subset of the above release

Data files are provided in zipped CSV and XLS formats and can be downloaded below. The XLS files have variable labels attached, but do not include all the possible rows from Employment Flows, due to size constraints.

State/Territory: All ▼

[View Files](#)



PSEO Help

Learn more about PSEO by choosing one of the links below.

- [PSEO Methodology and Data Sources](#)
- [PSEO Data Notices](#) (427 KB)
- [PSEO Data Schema for Most Recent Release](#)
- [PSEO Code Samples](#)
- [PSEO Technical Documentation](#) (213 KB)
- [Technical Appendix for PSEO Protection System](#) (225 KB)
- [PSEO Partnership SOPs](#)



Important Considerations

- Variables must be identified correctly to avoid errors.
- Consider how you want to filter data and using which variables.
- Consider whether it is possible no data will be retrieved.
- Must specify a geographic clause (e.g., for=us:1)

DEFAULTS

- DEGREE_LEVEL defaults to 05 (Bachelor's).
- GRAD_COHORT defaults to 0000 (All Cohorts).
- CIPCODE defaults to 00 (All Instructional Programs).



Best Practices

- Start simple.
 - Build your query by choosing one endpoint and narrowly targeted filters.
- Include labels.
 - Ensure comprehensibility by requesting label fields.
- Prepare for potentially null data.
 - Plan for what to do when no records are returned based on the selected filters.



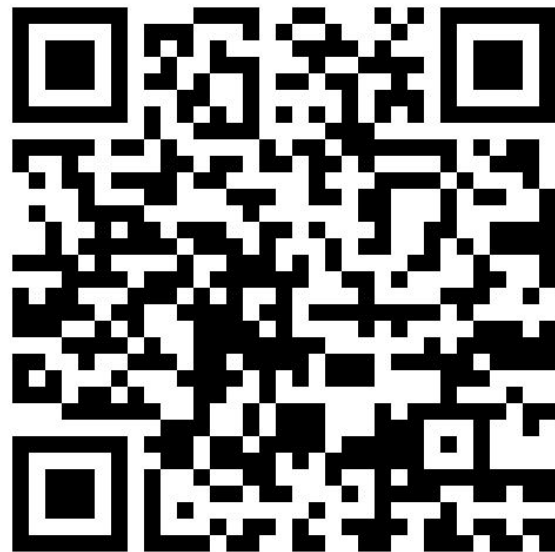
Python Demo

Feedback

**Please complete this brief survey on
the PSEOC Summer Webinar Series
2025.**

**We value your feedback on this
series and ideas for future
educational opportunities.**

Survey link



Coming Up

PSEO Coalition Virtual Showcase: Research & Dashboards Using PSEO Data

September 25, 2025

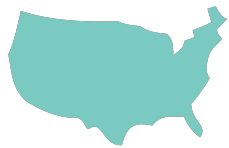
2:00 – 4:00 PM ET

<https://pseocoalition.org/events/>



PSEO Resources

PSEO Resources



PSEO Coalition

pseocoalition.org



Resource Library

pseocoalition.org/resource-library/



PSEO Explorer

lehd.ces.census.gov/applications/pseo



PSEO Datasets

lehd.ces.census.gov/data/pseo_experimental.html

