Reproducible Workflows with IPUMS CPS & the IPUMS API

Renae Rodgers

February 21, 2023

Zoom Logistics

1. Webinar is being recorded and will be posted
2. Real-time closed captions are being auto-generated
   - Turn on/off by clicking “CC” button in Zoom controls
3. Send questions about Zoom directly to the host (IPUMS)
4. Submit content questions with Q&A tool
5. We will post a written Q&A document following webinar

The Plan for Today's Webinar

1. Background on IPUMS CPS example
2. Sustainable Workflows
3. A little bit about APIs
4. Using the IPUMS API in our IPUMS CPS workflow

The Example

Comparing Poverty Measure in IPUMS CPS Over Time

The Task

We want to examine trends in poverty over time using both the supplemental poverty measure (SPM) and the official poverty measure (OPM). In this example we will

1. Calculate poverty rates using both SPM and OPM indicators for each year in our data
2. Visualize these rates over time

Some Background Information

- The Current Population Survey (CPS) is a monthly labor force survey
- The Annual Social and Economic Supplement (ASEC) is a special supplement to the CPS that is collected in February-April every year.
- The ASEC survey asks respondents questions about the previous calendar year.

More Background Information

- Poverty information is available only in the ASEC files.
- There are two different measure of poverty available in the ASEC data.
  - Supplemental Poverty Measure (SPM)
  - Official Poverty Measure (OPM)
- These measures differ in the unit of analysis, calculation of poverty thresholds, and included resources.

Workflows

What is a workflow?

What is a workflow?

“[…] a system of managing repetitive processes and tasks which occur in a particular order.” – IBM

“[…] and end-to-end process that helps teams meet their goals by connecting the right people to the right data at the right time” - asana

What is a workflow?

“To put it plainly, it’s the **method you create for getting your stuff done.**” - Trello

Why are sustainable workflows important?

Sustainable workflows…

1. are less error-prone!
2. are easier to debug when we do mess up!
3. Make collaborating easier!
   - with yourself
   - and others!

A note about tools…

Sustainable workflows can be set up using ANY programming language and ANY statistical software!

A sustainable workflow is…

1. Flexible

2. Organized

3. Reproducible

A sustainable workflow is…

1. Flexible
2. Organized
3. Reproducible

In a flexible workflow…

1. It is easy to alter or update inputs
   - For example, to apply the same set of steps in the same order to different months or years of IPUMS CPS data
2. It is easy to apply the same functionality to a different set of inputs
   - For example, calculating poverty rates using different poverty measures
3. It is easy to add or subtract steps or correct errors
   - For example, a mistake in a step that is applied multiple times needs to be fixed in only one place.

A sustainable workflow is…

1. Flexible
2. Organized
3. Reproducible

In an organized workflow…

1. Steps proceed in a logical order

2. Steps and functionality are well-documented
   - This can be as simple as annotating your code with comments!
   - Giving functions descriptive names helps too!

A sustainable workflow is…

1. Flexible
2. Organized
3. Reproducible

A reproducible workflow…

1. Given the same input(s), a reproducible workflow will produce the same output(s). Every time.

An Example IPUMS CPS Workflow

Trends in Poverty Rates Over Time

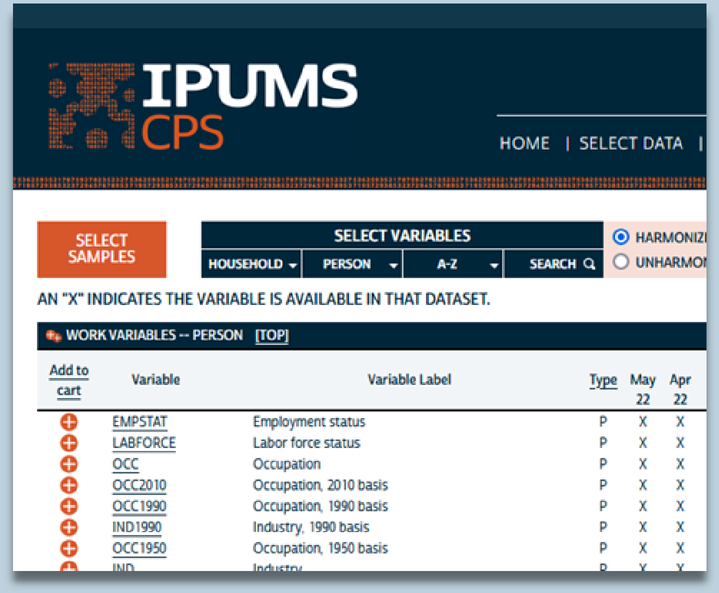
The Task

We want to examine trends in poverty over time using both the supplemental poverty measure (SPM) and the official poverty measure (OPM). In this example we will

1. Calculate poverty rates using both SPM and OPM indicators for each year in our data
2. Visualize these rates over time

Get the Data

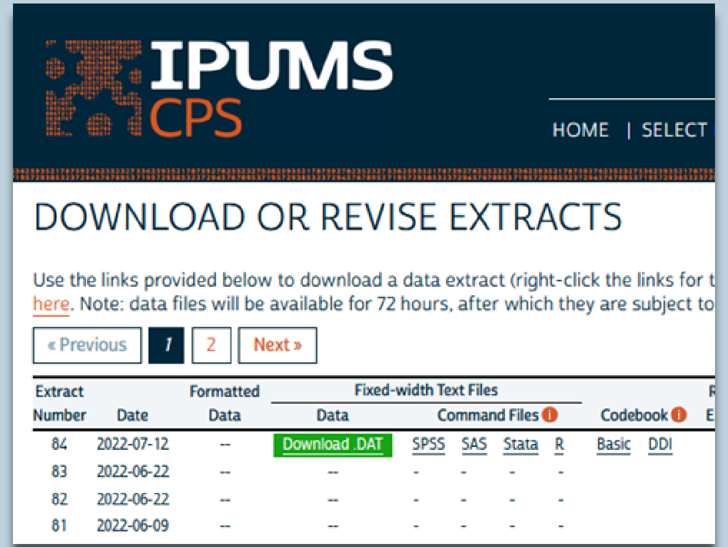
1. Go to cps.ipums.org
2. Use the website to select samples
   - 2010-2021 ASEC data
3. Use the website to select poverty vars
   - SPMPOV, SPMWT
   - OFFPOV, ASECWT
4. Wait for your extract to be processed
5. Download your extract file and accompanying metadata
6. Decompress the data file



AN "X" INDICATES THE VARIABLE IS AVAILABLE IN THAT DATASET.

WORK VARIABLES -- PERSON [TOP]

Add to cart	Variable	Variable Label	Type	May 22	Apr 22
⊕	EMPSTAT	Employment status	P	X	X
⊕	LABFORCE	Labor force status	P	X	X
⊕	OCC	Occupation	P	X	X
⊕	OCC2010	Occupation, 2010 basis	P	X	X
⊕	OCC1990	Occupation, 1990 basis	P	X	X
⊕	IND1990	Industry, 1990 basis	P	X	X
⊕	OCC1950	Occupation, 1950 basis	P	X	X
⊕	IND	Industry	P	X	X



DOWNLOAD OR REVISE EXTRACTS

Use the links provided below to download a data extract (right-click the links for t
here. Note: data files will be available for 72 hours, after which they are subject to

« Previous | 1 | 2 | Next »

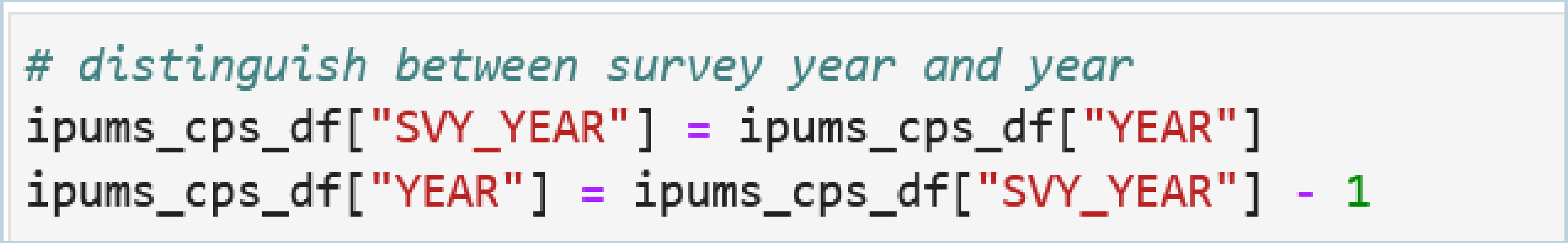
Extract		Formatted	Fixed-width Text Files							
Number	Date	Data	Data	Command Files				Codebook		
84	2022-07-12	--	Download .DAT	SPSS	SAS	Stata	R	Basic	DDI	
83	2022-06-22	--	--	-	-	-	-			
82	2022-06-22	--	--	-	-	-	-			
81	2022-06-09	--	--	-	-	-	-			

…read in the data

… using your programming language or stats package of choice

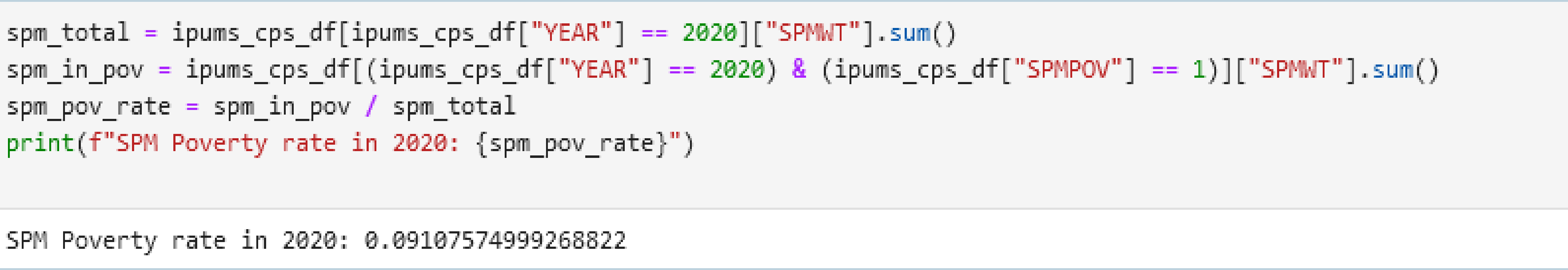
…clean up and organize the data

The ASEC survey asks respondents about the previous calendar year, so data collected in the survey year 2021 describes the year 2020.



```python
# distinguish between survey year and year
ipums_cps_df["SVY_YEAR"] = ipums_cps_df["YEAR"]
ipums_cps_df["YEAR"] = ipums_cps_df["SVY_YEAR"] - 1
```

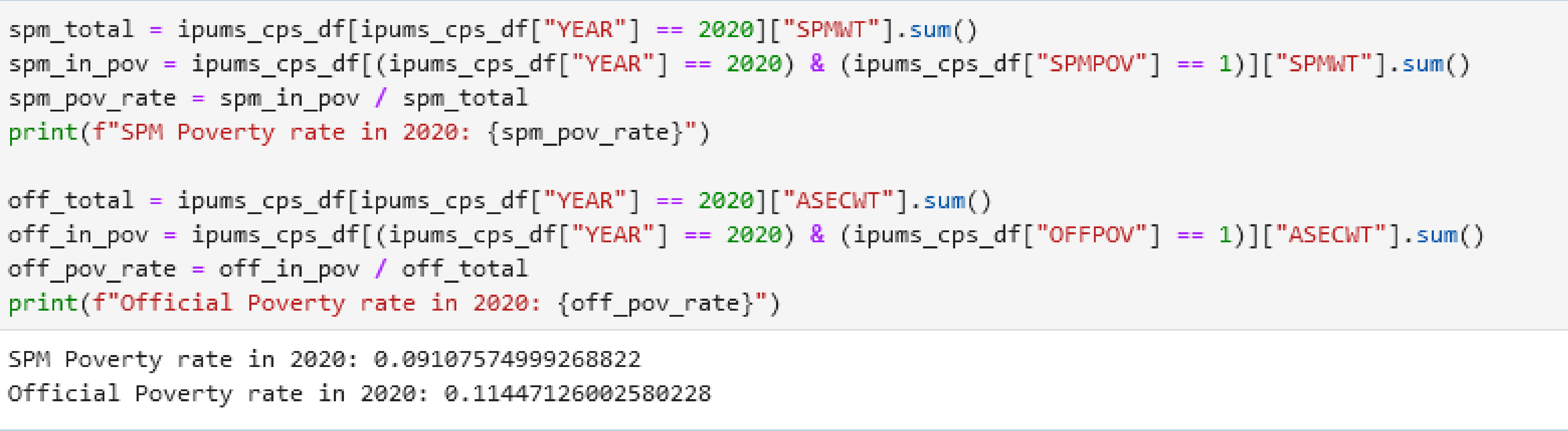
…calculate poverty rates for each year



```python
spm_total = ipums_cps_df[ipums_cps_df["YEAR"] == 2020]["SPMWT"].sum()
spm_in_pov = ipums_cps_df[(ipums_cps_df["YEAR"] == 2020) & (ipums_cps_df["SPMPOV"] == 1)]["SPMWT"].sum()
spm_pov_rate = spm_in_pov / spm_total
print(f"SPM Poverty rate in 2020: {spm_pov_rate}")
```

```
SPM Poverty rate in 2020: 0.09107574999268822
```

…calculate poverty rates for each year

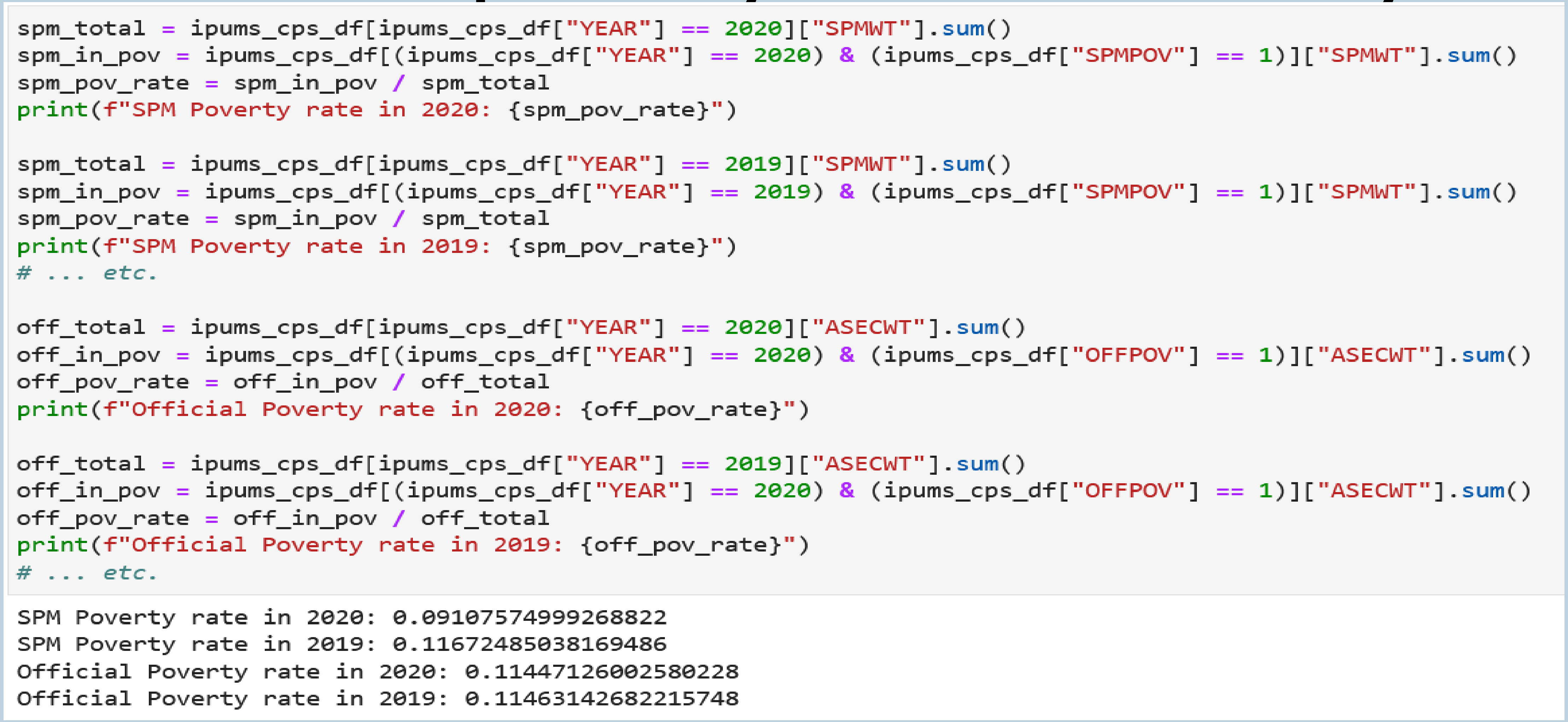


```python
spm_total = ipums_cps_df[ipums_cps_df["YEAR"] == 2020]["SPMWT"].sum()
spm_in_pov = ipums_cps_df[(ipums_cps_df["YEAR"] == 2020) & (ipums_cps_df["SPMPOV"] == 1)]["SPMWT"].sum()
spm_pov_rate = spm_in_pov / spm_total
print(f"SPM Poverty rate in 2020: {spm_pov_rate}")

off_total = ipums_cps_df[ipums_cps_df["YEAR"] == 2020]["ASECWT"].sum()
off_in_pov = ipums_cps_df[(ipums_cps_df["YEAR"] == 2020) & (ipums_cps_df["OFFPOV"] == 1)]["ASECWT"].sum()
off_pov_rate = off_in_pov / off_total
print(f"Official Poverty rate in 2020: {off_pov_rate}")
```

```
SPM Poverty rate in 2020: 0.09107574999268822
Official Poverty rate in 2020: 0.11447126002580228
```

IPUMS

...calculate poverty rates for each year



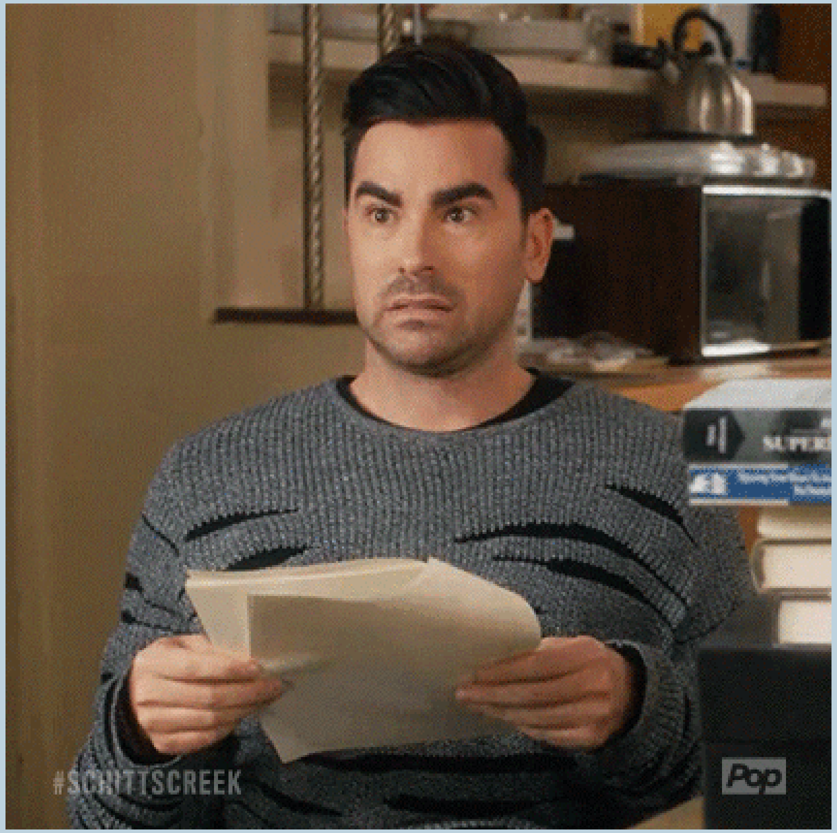
```python
spm_total = ipums_cps_df[ipums_cps_df["YEAR"] == 2020]["SPMWT"].sum()
spm_in_pov = ipums_cps_df[(ipums_cps_df["YEAR"] == 2020) & (ipums_cps_df["SPMPOV"] == 1)]["SPMWT"].sum()
spm_pov_rate = spm_in_pov / spm_total
print(f"SPM Poverty rate in 2020: {spm_pov_rate}")

spm_total = ipums_cps_df[ipums_cps_df["YEAR"] == 2019]["SPMWT"].sum()
spm_in_pov = ipums_cps_df[(ipums_cps_df["YEAR"] == 2019) & (ipums_cps_df["SPMPOV"] == 1)]["SPMWT"].sum()
spm_pov_rate = spm_in_pov / spm_total
print(f"SPM Poverty rate in 2019: {spm_pov_rate}")
# ... etc.

off_total = ipums_cps_df[ipums_cps_df["YEAR"] == 2020]["ASECWT"].sum()
off_in_pov = ipums_cps_df[(ipums_cps_df["YEAR"] == 2020) & (ipums_cps_df["OFFPOV"] == 1)]["ASECWT"].sum()
off_pov_rate = off_in_pov / off_total
print(f"Official Poverty rate in 2020: {off_pov_rate}")

off_total = ipums_cps_df[ipums_cps_df["YEAR"] == 2019]["ASECWT"].sum()
off_in_pov = ipums_cps_df[(ipums_cps_df["YEAR"] == 2020) & (ipums_cps_df["OFFPOV"] == 1)]["ASECWT"].sum()
off_pov_rate = off_in_pov / off_total
print(f"Official Poverty rate in 2019: {off_pov_rate}")
# ... etc.
```

```
SPM Poverty rate in 2020: 0.09107574999268822
SPM Poverty rate in 2019: 0.11672485038169486
Official Poverty rate in 2020: 0.11447126002580228
Official Poverty rate in 2019: 0.11463142682215748
```

IPUMS

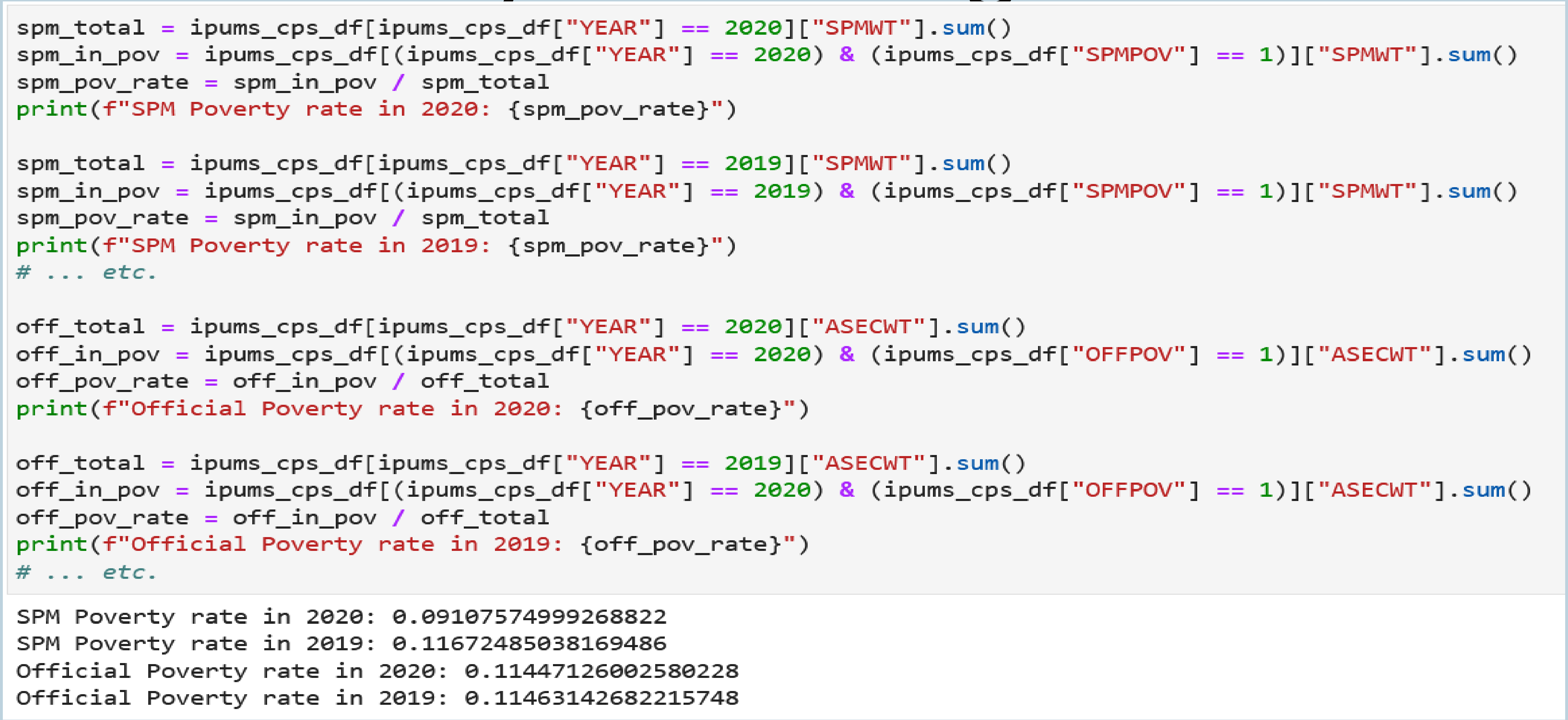
…calculate poverty rates for each year

…This is becoming awkward



And this violates all of our rules!

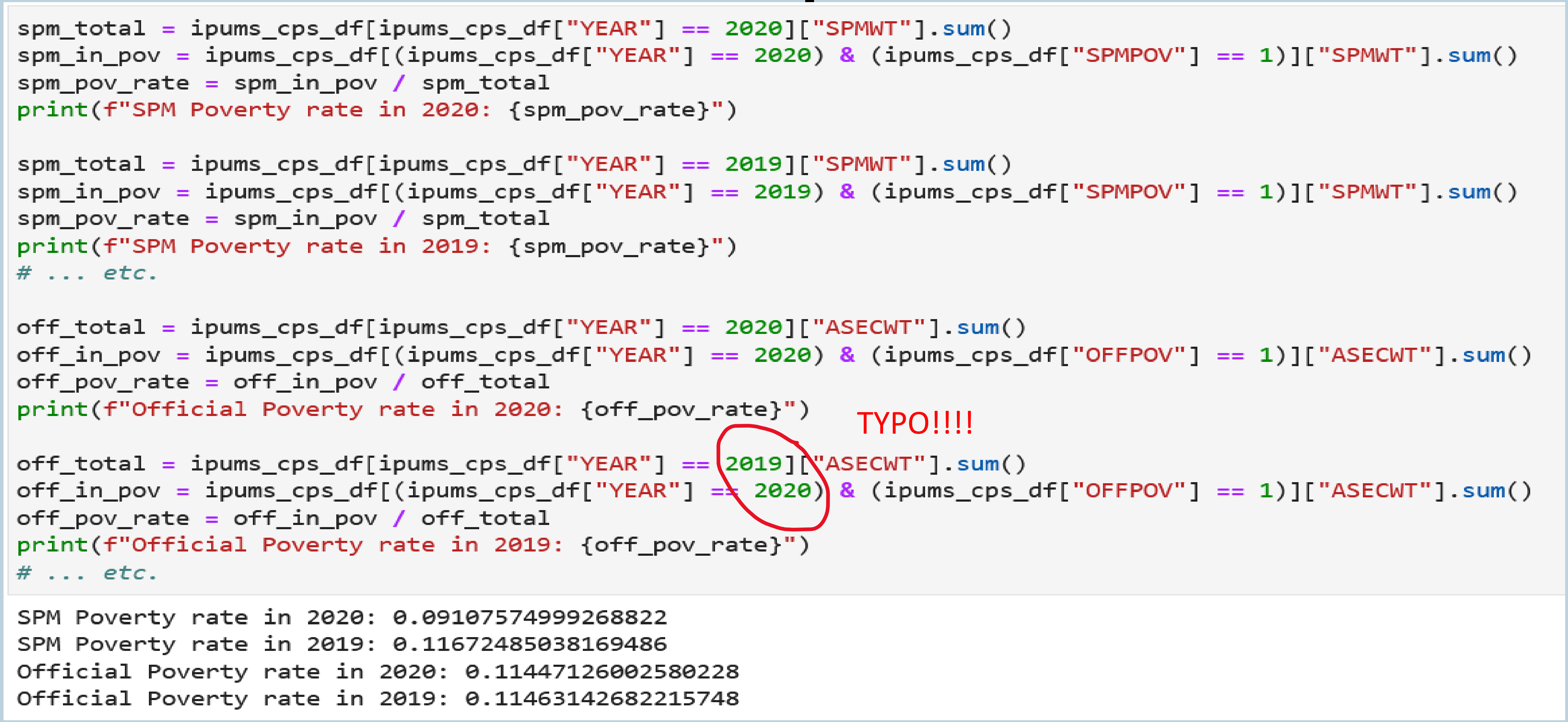
Why is this not good?



```
spm_total = ipums_cps_df[ipums_cps_df["YEAR"] == 2020]["SPMWT"].sum()
spm_in_pov = ipums_cps_df[(ipums_cps_df["YEAR"] == 2020) & (ipums_cps_df["SPMPOV"] == 1)]["SPMWT"].sum()
spm_pov_rate = spm_in_pov / spm_total
print(f"SPM Poverty rate in 2020: {spm_pov_rate}")

spm_total = ipums_cps_df[ipums_cps_df["YEAR"] == 2019]["SPMWT"].sum()
spm_in_pov = ipums_cps_df[(ipums_cps_df["YEAR"] == 2019) & (ipums_cps_df["SPMPOV"] == 1)]["SPMWT"].sum()
spm_pov_rate = spm_in_pov / spm_total
print(f"SPM Poverty rate in 2019: {spm_pov_rate}")
# ... etc.

off_total = ipums_cps_df[ipums_cps_df["YEAR"] == 2020]["ASECWT"].sum()
off_in_pov = ipums_cps_df[(ipums_cps_df["YEAR"] == 2020) & (ipums_cps_df["OFFPOV"] == 1)]["ASECWT"].sum()
off_pov_rate = off_in_pov / off_total
print(f"Official Poverty rate in 2020: {off_pov_rate}")

off_total = ipums_cps_df[ipums_cps_df["YEAR"] == 2019]["ASECWT"].sum()
off_in_pov = ipums_cps_df[(ipums_cps_df["YEAR"] == 2020) & (ipums_cps_df["OFFPOV"] == 1)]["ASECWT"].sum()
off_pov_rate = off_in_pov / off_total
print(f"Official Poverty rate in 2019: {off_pov_rate}")
# ... etc.
```

```
SPM Poverty rate in 2020: 0.09107574999268822
SPM Poverty rate in 2019: 0.11672485038169486
Official Poverty rate in 2020: 0.11447126002580228
Official Poverty rate in 2019: 0.11463142682215748
```

1. Error-prone!!



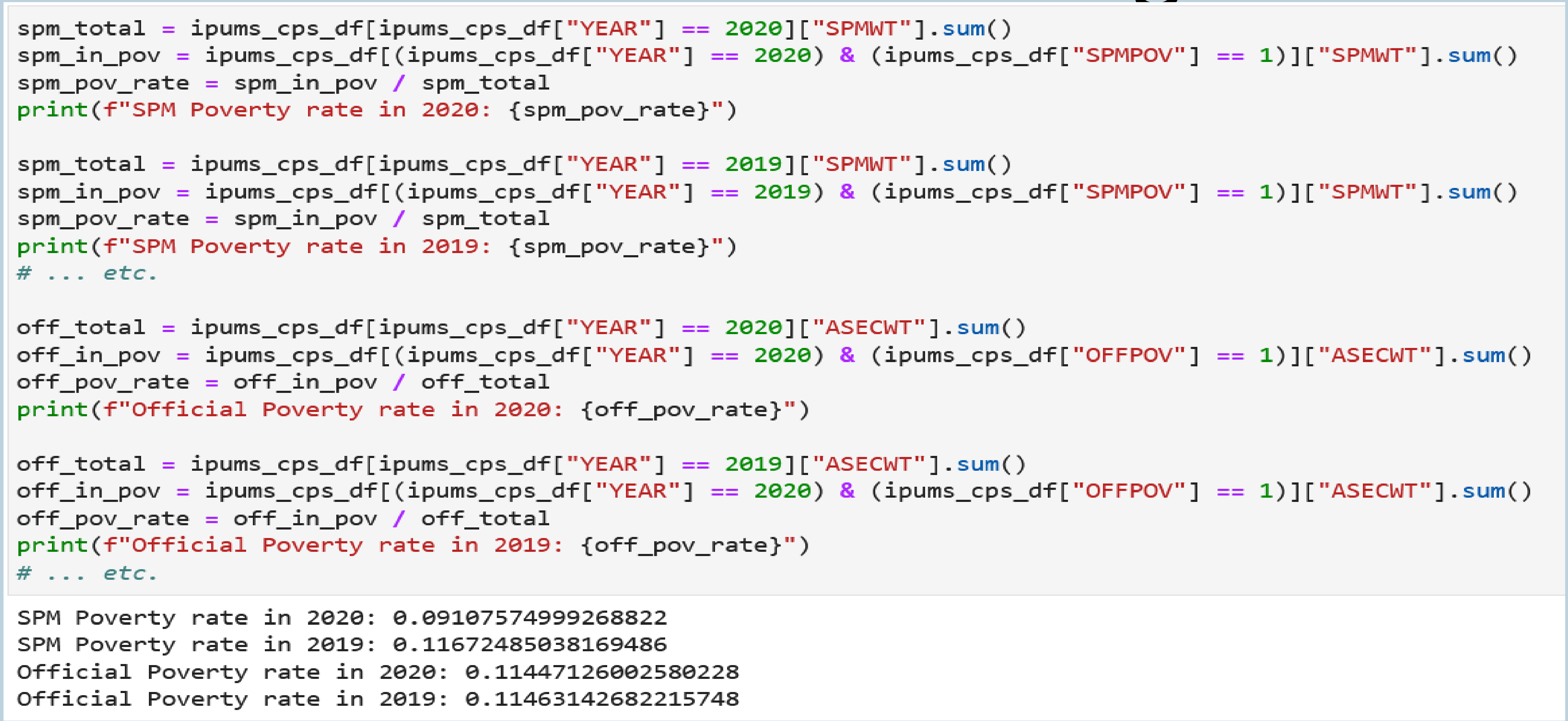
```python
spm_total = ipums_cps_df[ipums_cps_df["YEAR"] == 2020]["SPMWT"].sum()
spm_in_pov = ipums_cps_df[(ipums_cps_df["YEAR"] == 2020) & (ipums_cps_df["SPMPOV"] == 1)]["SPMWT"].sum()
spm_pov_rate = spm_in_pov / spm_total
print(f"SPM Poverty rate in 2020: {spm_pov_rate}")

spm_total = ipums_cps_df[ipums_cps_df["YEAR"] == 2019]["SPMWT"].sum()
spm_in_pov = ipums_cps_df[(ipums_cps_df["YEAR"] == 2019) & (ipums_cps_df["SPMPOV"] == 1)]["SPMWT"].sum()
spm_pov_rate = spm_in_pov / spm_total
print(f"SPM Poverty rate in 2019: {spm_pov_rate}")
# ... etc.

off_total = ipums_cps_df[ipums_cps_df["YEAR"] == 2020]["ASECWT"].sum()
off_in_pov = ipums_cps_df[(ipums_cps_df["YEAR"] == 2020) & (ipums_cps_df["OFFPOV"] == 1)]["ASECWT"].sum()
off_pov_rate = off_in_pov / off_total
print(f"Official Poverty rate in 2020: {off_pov_rate}")

off_total = ipums_cps_df[ipums_cps_df["YEAR"] == 2019]["ASECWT"].sum()
off_in_pov = ipums_cps_df[(ipums_cps_df["YEAR"] == 2020) & (ipums_cps_df["OFFPOV"] == 1)]["ASECWT"].sum()
off_pov_rate = off_in_pov / off_total
print(f"Official Poverty rate in 2019: {off_pov_rate}")
# ... etc.
```

TYPO!!!!

```
SPM Poverty rate in 2020: 0.09107574999268822
SPM Poverty rate in 2019: 0.11672485038169486
Official Poverty rate in 2020: 0.11447126002580228
Official Poverty rate in 2019: 0.11463142682215748
```

IPUMS

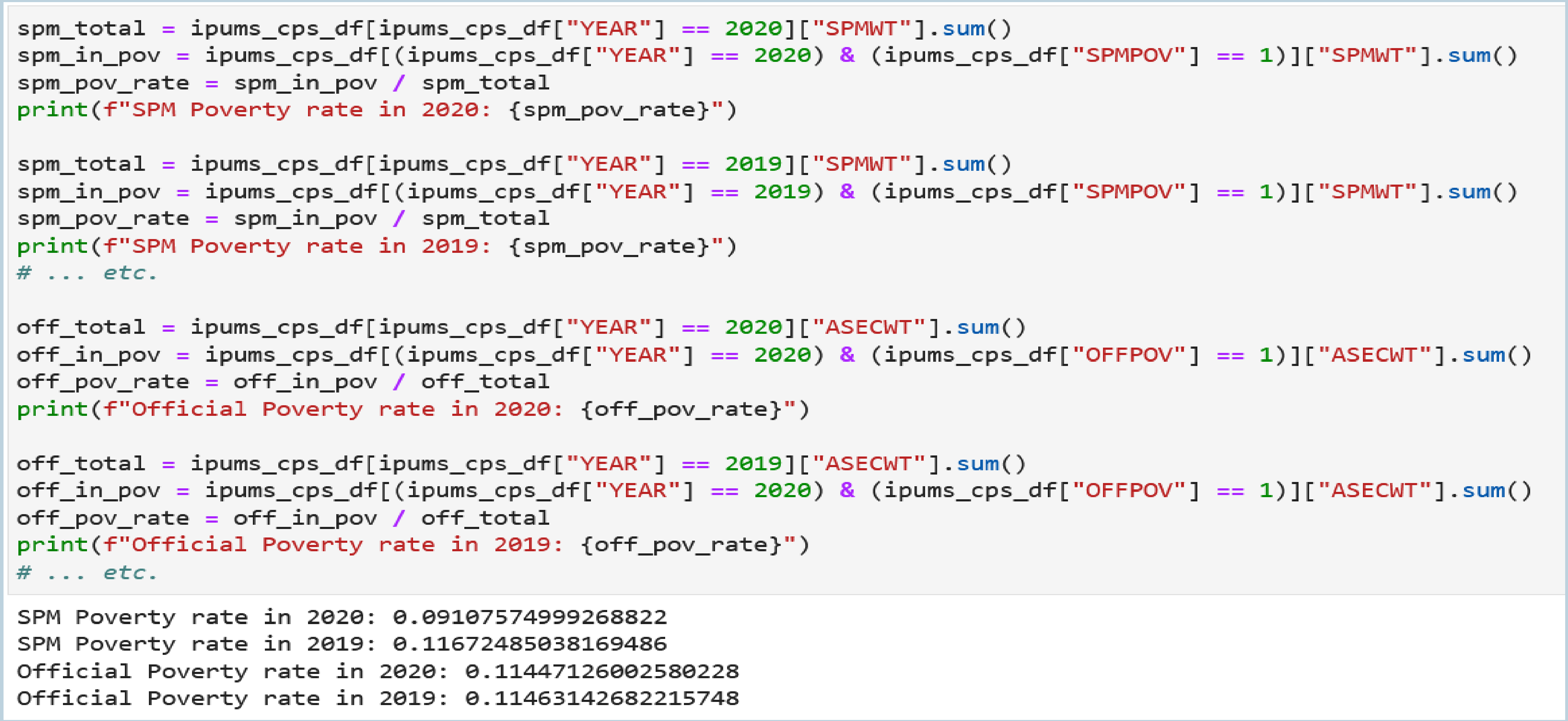
2. Difficult to debug



```python
spm_total = ipums_cps_df[ipums_cps_df["YEAR"] == 2020]["SPMWT"].sum()
spm_in_pov = ipums_cps_df[(ipums_cps_df["YEAR"] == 2020) & (ipums_cps_df["SPMPOV"] == 1)]["SPMWT"].sum()
spm_pov_rate = spm_in_pov / spm_total
print(f"SPM Poverty rate in 2020: {spm_pov_rate}")

spm_total = ipums_cps_df[ipums_cps_df["YEAR"] == 2019]["SPMWT"].sum()
spm_in_pov = ipums_cps_df[(ipums_cps_df["YEAR"] == 2019) & (ipums_cps_df["SPMPOV"] == 1)]["SPMWT"].sum()
spm_pov_rate = spm_in_pov / spm_total
print(f"SPM Poverty rate in 2019: {spm_pov_rate}")
# ... etc.

off_total = ipums_cps_df[ipums_cps_df["YEAR"] == 2020]["ASECWT"].sum()
off_in_pov = ipums_cps_df[(ipums_cps_df["YEAR"] == 2020) & (ipums_cps_df["OFFPOV"] == 1)]["ASECWT"].sum()
off_pov_rate = off_in_pov / off_total
print(f"Official Poverty rate in 2020: {off_pov_rate}")

off_total = ipums_cps_df[ipums_cps_df["YEAR"] == 2019]["ASECWT"].sum()
off_in_pov = ipums_cps_df[(ipums_cps_df["YEAR"] == 2020) & (ipums_cps_df["OFFPOV"] == 1)]["ASECWT"].sum()
off_pov_rate = off_in_pov / off_total
print(f"Official Poverty rate in 2019: {off_pov_rate}")
# ... etc.
```

```
SPM Poverty rate in 2020: 0.09107574999268822
SPM Poverty rate in 2019: 0.11672485038169486
Official Poverty rate in 2020: 0.11447126002580228
Official Poverty rate in 2019: 0.11463142682215748
```

3. Unclear



```python
spm_total = ipums_cps_df[ipums_cps_df["YEAR"] == 2020]["SPMWT"].sum()
spm_in_pov = ipums_cps_df[(ipums_cps_df["YEAR"] == 2020) & (ipums_cps_df["SPMPOV"] == 1)]["SPMWT"].sum()
spm_pov_rate = spm_in_pov / spm_total
print(f"SPM Poverty rate in 2020: {spm_pov_rate}")

spm_total = ipums_cps_df[ipums_cps_df["YEAR"] == 2019]["SPMWT"].sum()
spm_in_pov = ipums_cps_df[(ipums_cps_df["YEAR"] == 2019) & (ipums_cps_df["SPMPOV"] == 1)]["SPMWT"].sum()
spm_pov_rate = spm_in_pov / spm_total
print(f"SPM Poverty rate in 2019: {spm_pov_rate}")
# ... etc.

off_total = ipums_cps_df[ipums_cps_df["YEAR"] == 2020]["ASECWT"].sum()
off_in_pov = ipums_cps_df[(ipums_cps_df["YEAR"] == 2020) & (ipums_cps_df["OFFPOV"] == 1)]["ASECWT"].sum()
off_pov_rate = off_in_pov / off_total
print(f"Official Poverty rate in 2020: {off_pov_rate}")

off_total = ipums_cps_df[ipums_cps_df["YEAR"] == 2019]["ASECWT"].sum()
off_in_pov = ipums_cps_df[(ipums_cps_df["YEAR"] == 2020) & (ipums_cps_df["OFFPOV"] == 1)]["ASECWT"].sum()
off_pov_rate = off_in_pov / off_total
print(f"Official Poverty rate in 2019: {off_pov_rate}")
# ... etc.
```

```
SPM Poverty rate in 2020: 0.09107574999268822
SPM Poverty rate in 2019: 0.11672485038169486
Official Poverty rate in 2020: 0.11447126002580228
Official Poverty rate in 2019: 0.11463142682215748
```

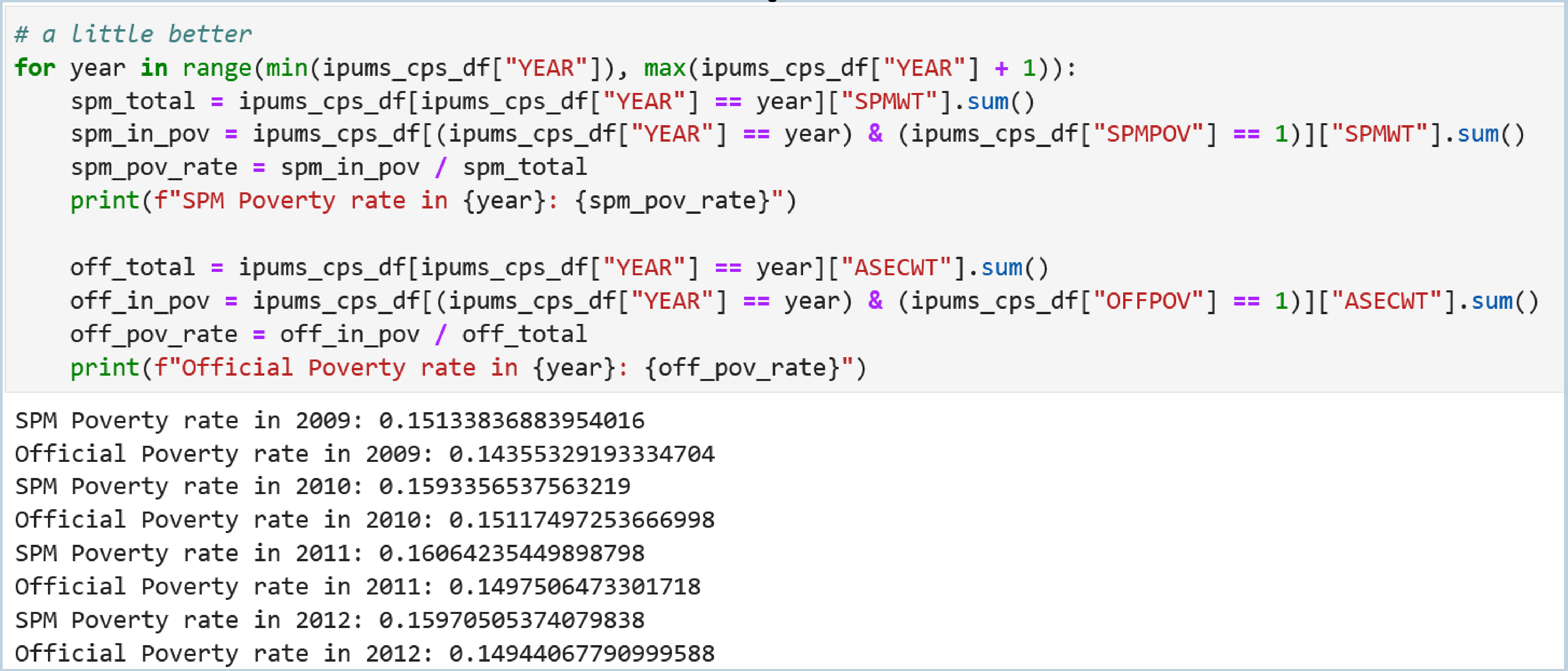
How can we make this better?

1. Use a for loop instead of copying-and-pasting

- Perform the logic for each value in a list of values in turn.

- In our example, loop over each year in our extract and calculate SPM and OPM poverty rates

A for-loop solution



```python
# a little better
for year in range(min(ipums_cps_df["YEAR"]), max(ipums_cps_df["YEAR"] + 1)):
    spm_total = ipums_cps_df[ipums_cps_df["YEAR"] == year]["SPMWT"].sum()
    spm_in_pov = ipums_cps_df[(ipums_cps_df["YEAR"] == year) & (ipums_cps_df["SPMPOV"] == 1)]["SPMWT"].sum()
    spm_pov_rate = spm_in_pov / spm_total
    print(f"SPM Poverty rate in {year}: {spm_pov_rate}")

    off_total = ipums_cps_df[ipums_cps_df["YEAR"] == year]["ASECWT"].sum()
    off_in_pov = ipums_cps_df[(ipums_cps_df["YEAR"] == year) & (ipums_cps_df["OFFPOV"] == 1)]["ASECWT"].sum()
    off_pov_rate = off_in_pov / off_total
    print(f"Official Poverty rate in {year}: {off_pov_rate}")
```

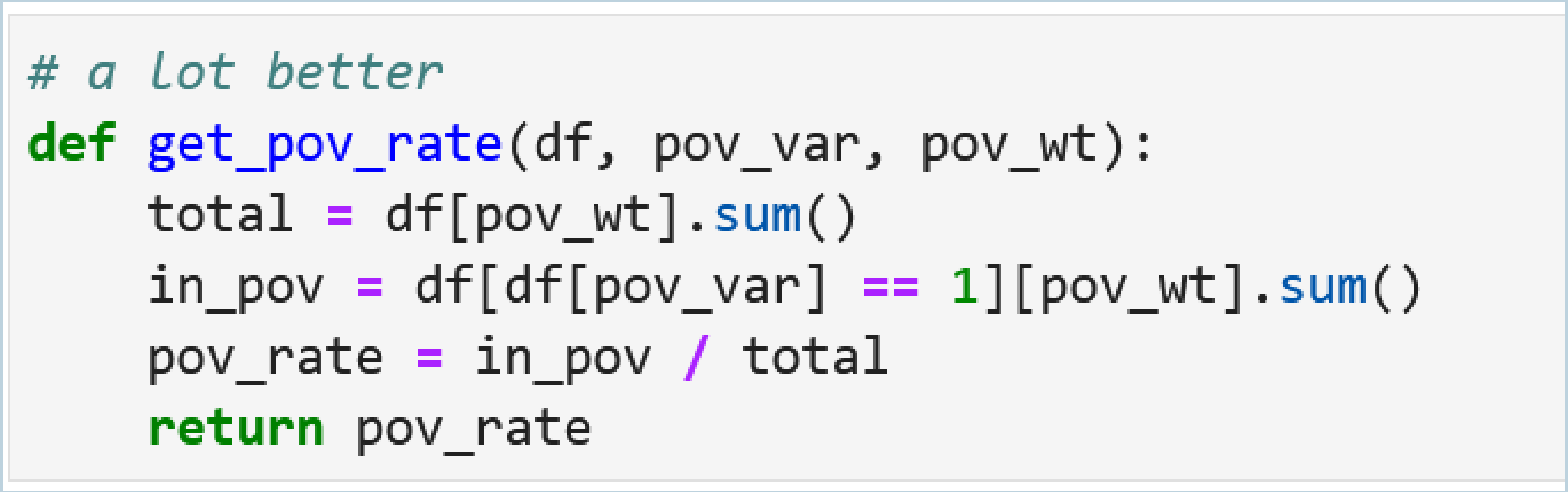
```
SPM Poverty rate in 2009: 0.15133836883954016
Official Poverty rate in 2009: 0.14355329193334704
SPM Poverty rate in 2010: 0.1593356537563219
Official Poverty rate in 2010: 0.15117497253666998
SPM Poverty rate in 2011: 0.16064235449898798
Official Poverty rate in 2011: 0.1497506473301718
SPM Poverty rate in 2012: 0.15970505374079838
Official Poverty rate in 2012: 0.14944067790999588
```

IPUMS

How can we make this better?

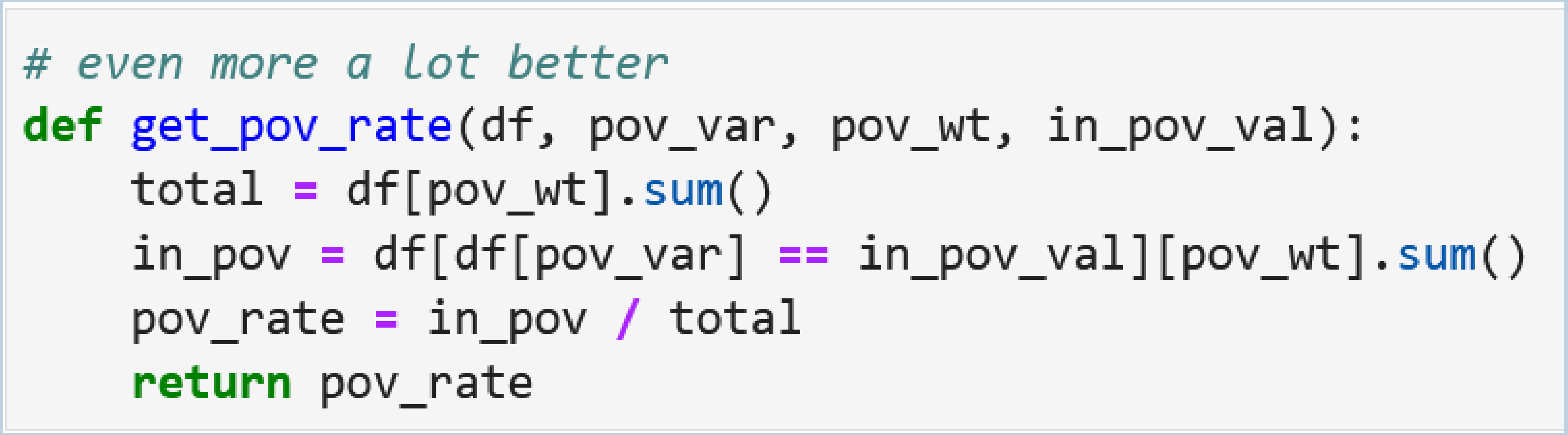
1. Put this logic for calculating poverty rates into a function
    - Easily allow multiple parameters to change
        - In our example, poverty variable and weight
    - Get better output
        - A data frame of poverty rates by year instead of a series of print statements
    - More readable code

A function solution



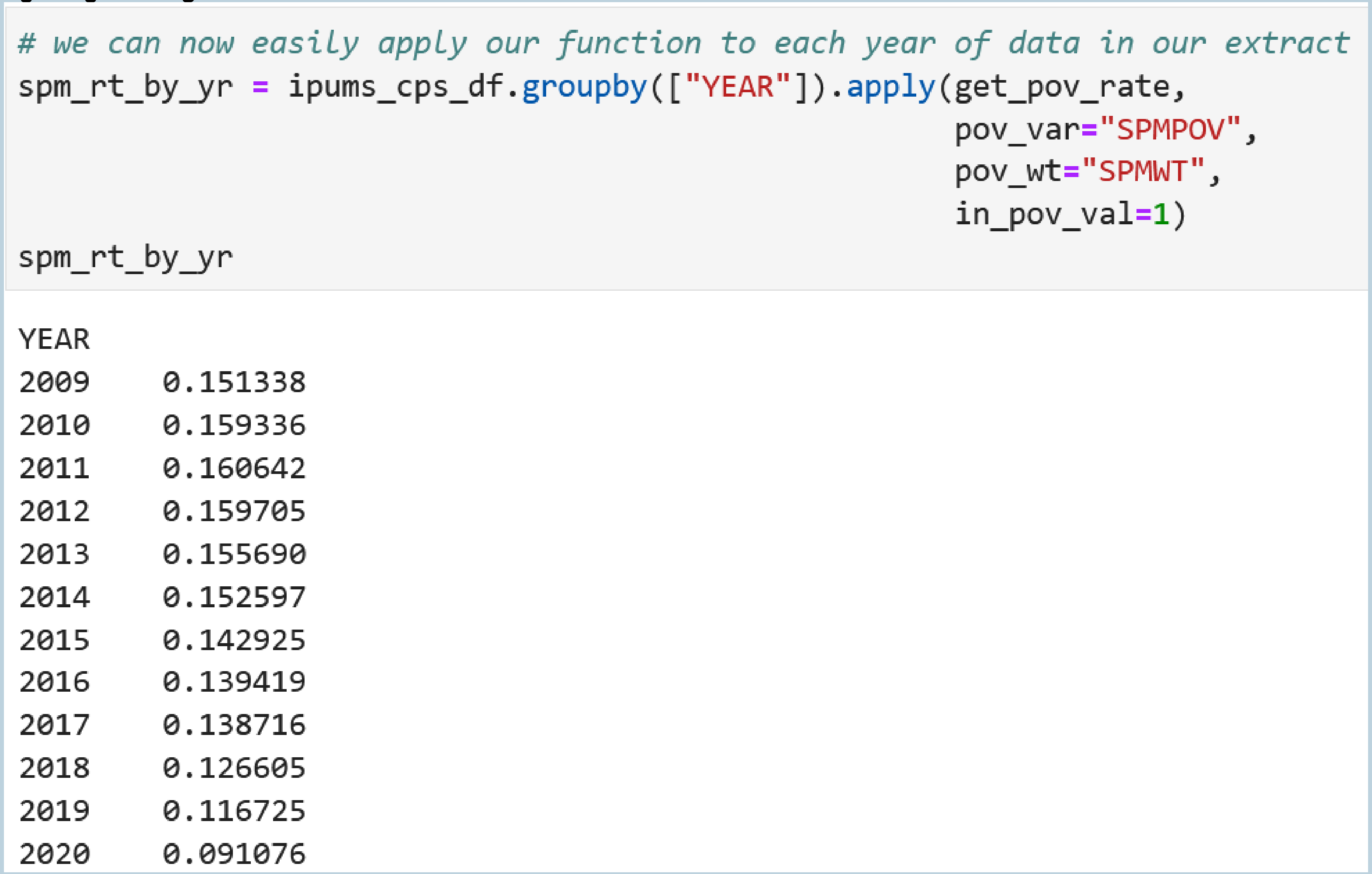
```python
# a lot better
def get_pov_rate(df, pov_var, pov_wt):
    total = df[pov_wt].sum()
    in_pov = df[df[pov_var] == 1][pov_wt].sum()
    pov_rate = in_pov / total
    return pov_rate
```

Another function solution



```python
# even more a lot better
def get_pov_rate(df, pov_var, pov_wt, in_pov_val):
    total = df[pov_wt].sum()
    in_pov = df[df[pov_var] == in_pov_val][pov_wt].sum()
    pov_rate = in_pov / total
    return pov_rate
```

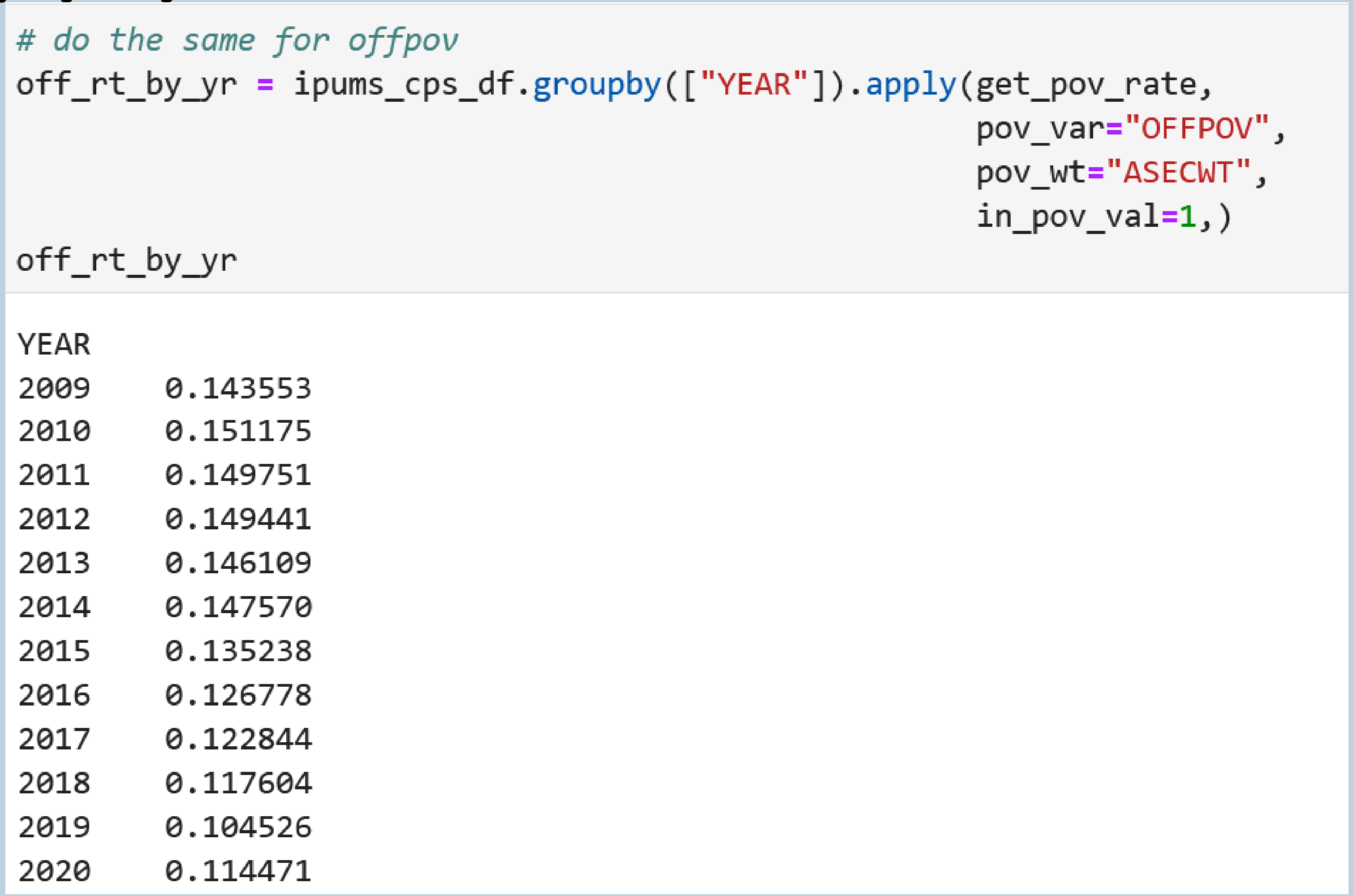
Apply the function to the data



```
# we can now easily apply our function to each year of data in our extract
spm_rt_by_yr = ipums_cps_df.groupby(["YEAR"]).apply(get_pov_rate,
                                                     pov_var="SPMPOV",
                                                     pov_wt="SPMWT",
                                                     in_pov_val=1)

spm_rt_by_yr
```

```
YEAR
2009    0.151338
2010    0.159336
2011    0.160642
2012    0.159705
2013    0.155690
2014    0.152597
2015    0.142925
2016    0.139419
2017    0.138716
2018    0.126605
2019    0.116725
2020    0.091076
```

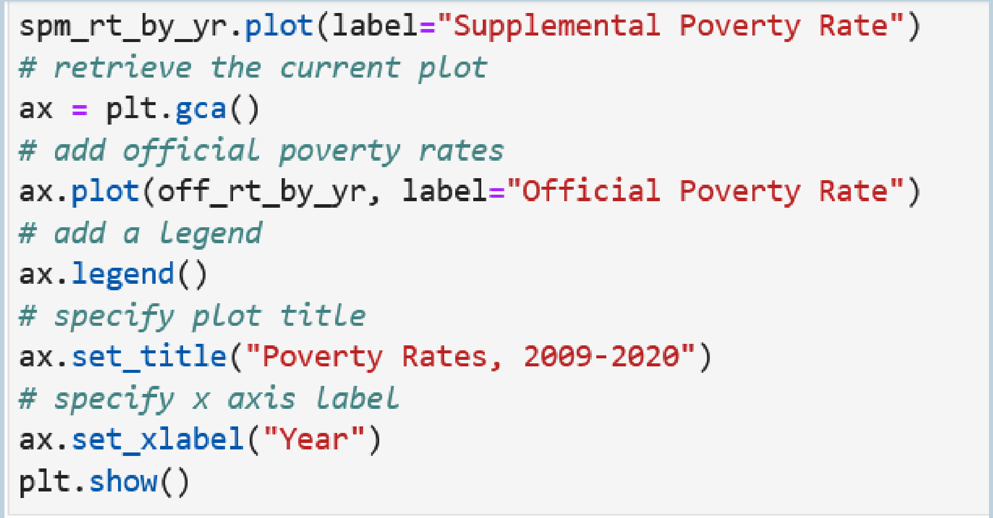
Apply the function to the data



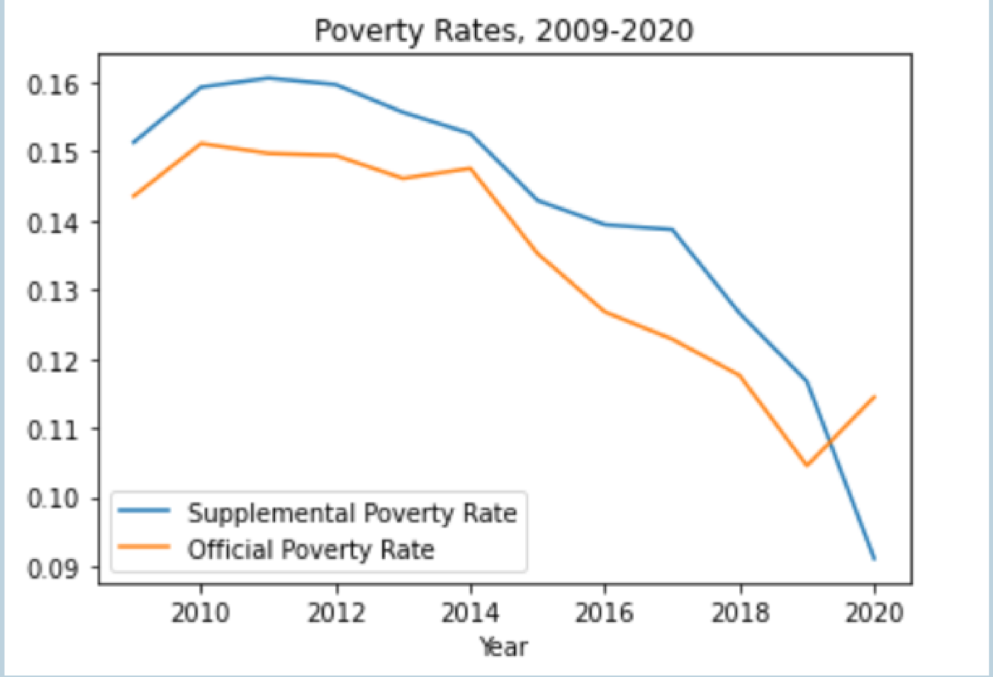
```
# do the same for offpov
off_rt_by_yr = ipums_cps_df.groupby(["YEAR"]).apply(get_pov_rate,
                                                     pov_var="OFFPOV",
                                                     pov_wt="ASECWT",
                                                     in_pov_val=1,)
off_rt_by_yr
```

```
YEAR
2009    0.143553
2010    0.151175
2011    0.149751
2012    0.149441
2013    0.146109
2014    0.147570
2015    0.135238
2016    0.126778
2017    0.122844
2018    0.117604
2019    0.104526
2020    0.114471
```

Plot it



```
spm_rt_by_yr.plot(label="Supplemental Poverty Rate")
# retrieve the current plot
ax = plt.gca()
# add official poverty rates
ax.plot(off_rt_by_yr, label="Official Poverty Rate")
# add a legend
ax.legend()
# specify plot title
ax.set_title("Poverty Rates, 2009-2020")
# specify x axis label
ax.set_xlabel("Year")
plt.show()
```



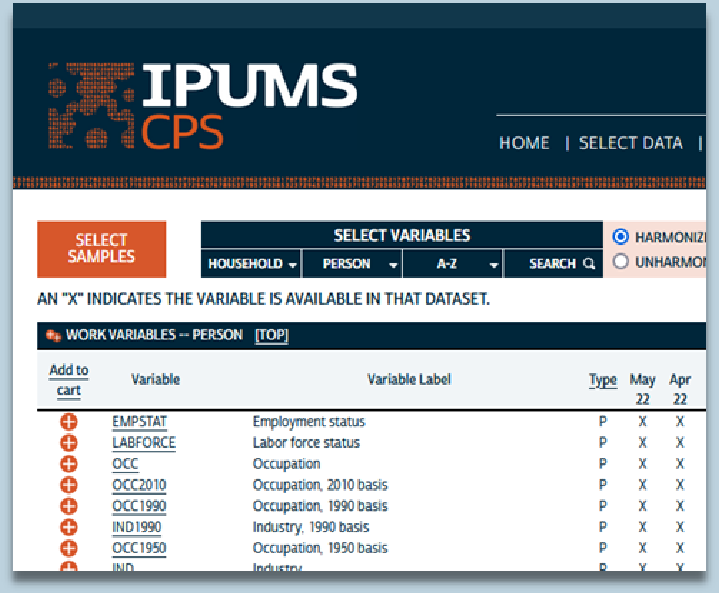
Cleaner, more readable, less error-prone code!

So…what about APIs?

Level-up your workflow for fun! (and maybe profit?)

Get the Data

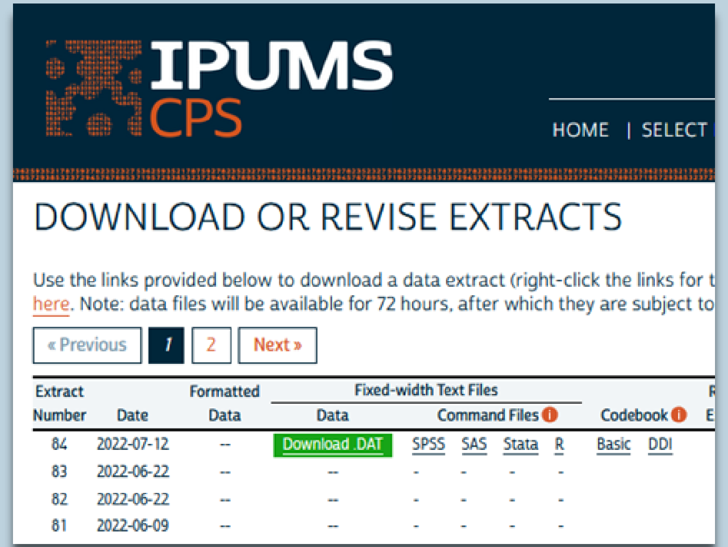
1. Go to cps.ipums.org
2. Use the website to select samples
   - 2010-2021 ASEC data
3. Use the website to select poverty vars
   - SPMPOV, SPMWT
   - OFFPOV, ASECWT
4. Wait for your extract to be processed
5. Download your extract file and accompanying metadata
6. Decompress the data file



AN "X" INDICATES THE VARIABLE IS AVAILABLE IN THAT DATASET.

WORK VARIABLES -- PERSON [TOP]

Add to cart	Variable	Variable Label	Type	May 22	Apr 22
⊕	EMPSTAT	Employment status	P	X	X
⊕	LABFORCE	Labor force status	P	X	X
⊕	OCC	Occupation	P	X	X
⊕	OCC2010	Occupation, 2010 basis	P	X	X
⊕	OCC1990	Occupation, 1990 basis	P	X	X
⊕	IND1990	Industry, 1990 basis	P	X	X
⊕	OCC1950	Occupation, 1950 basis	P	X	X
⊕	IND	Industry	P	X	X



DOWNLOAD OR REVISE EXTRACTS

Use the links provided below to download a data extract (right-click the links for t
here. Note: data files will be available for 72 hours, after which they are subject to

« Previous | 1 | 2 | Next »

Extract Number	Date	Formatted Data	Fixed-width Text Files Data	Command Files				Codebook	
84	2022-07-12	--	Download .DAT	SPSS	SAS	Stata	R	Basic	DDI
83	2022-06-22	--	--	-	-	-	-		
82	2022-06-22	--	--	-	-	-	-		
81	2022-06-09	--	--	-	-	-	-		

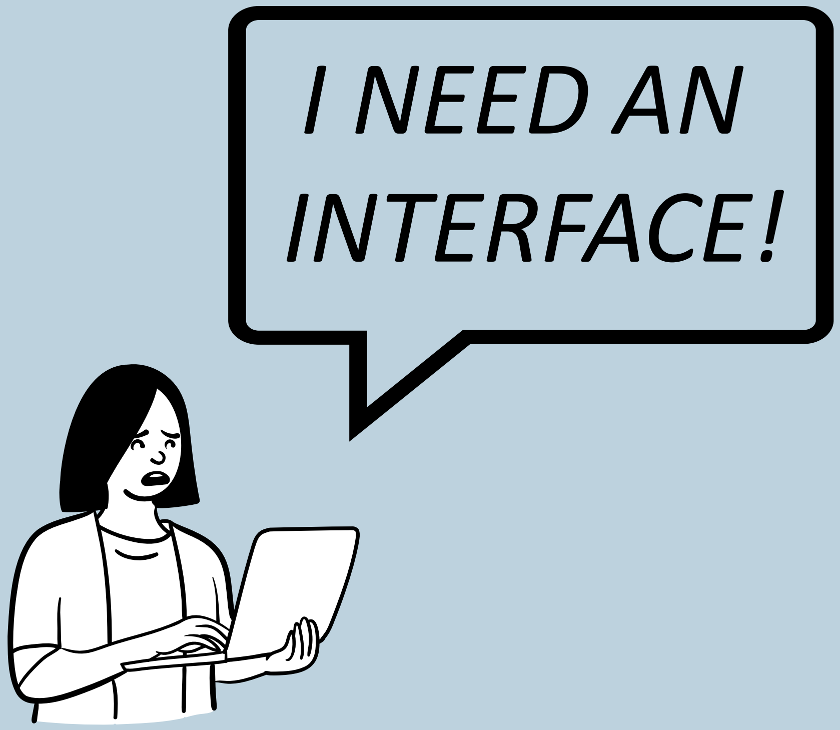
IPUMS

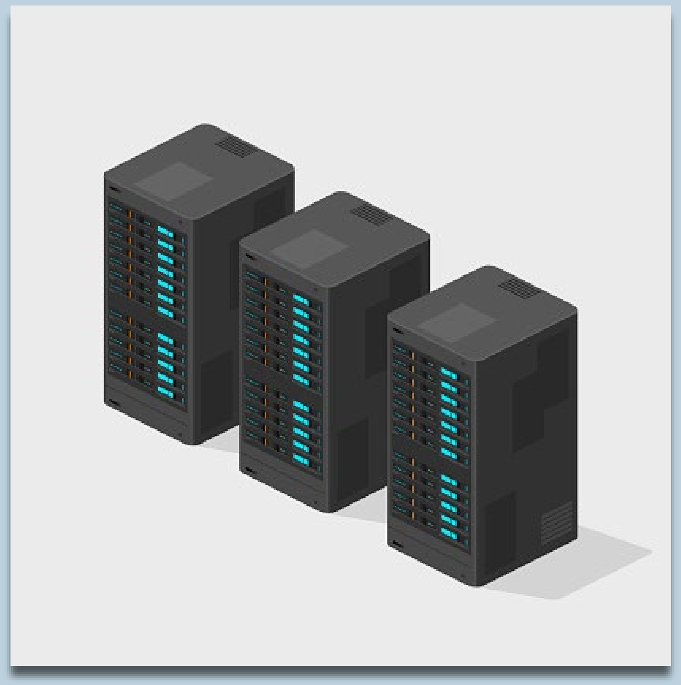
What is an API?

$$\underbrace{\text{Application Programming}}_{\text{using }\textbf{code}}\ \underbrace{\text{Interface}}_{\text{a way to }\textbf{interact}}$$

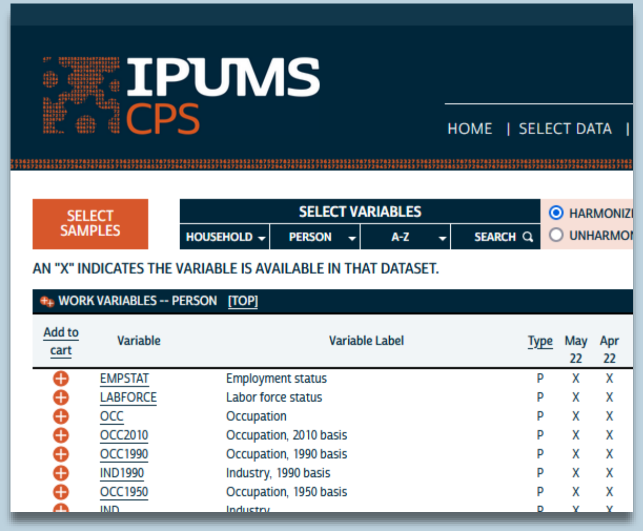
What is an API?

a way to **interact** using **code**





IPUMS



IPUMS CPS

HOME | SELECT DATA |

SELECT SAMPLES

SELECT VARIABLES

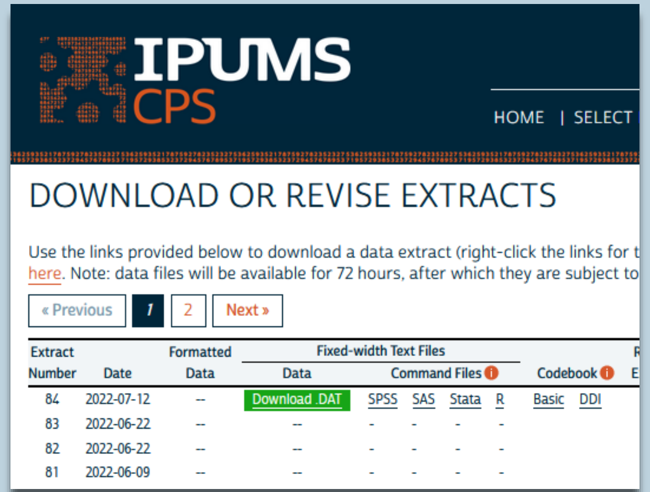
HOUSEHOLD | PERSON | A-Z | SEARCH

[✓ HARMONIZ] UNHARMON

AN "X" INDICATES THE VARIABLE IS AVAILABLE IN THAT DATASET.

WORK VARIABLES -- PERSON [TOP]

Add to cart	Variable	Variable Label	Type	May 22	Apr 22
⊕	EMPSTAT	Employment status	P	X	X
⊕	LABFORCE	Labor force status	P	X	X
⊕	OCC	Occupation	P	X	X
⊕	OCC2010	Occupation, 2010 basis	P	X	X
⊕	OCC1990	Occupation, 1990 basis	P	X	X
⊕	IND1990	Industry, 1990 basis	P	X	X
⊕	OCC1950	Occupation, 1950 basis	P	X	X
⊕	IND	Industry	P	X	X



IPUMS CPS

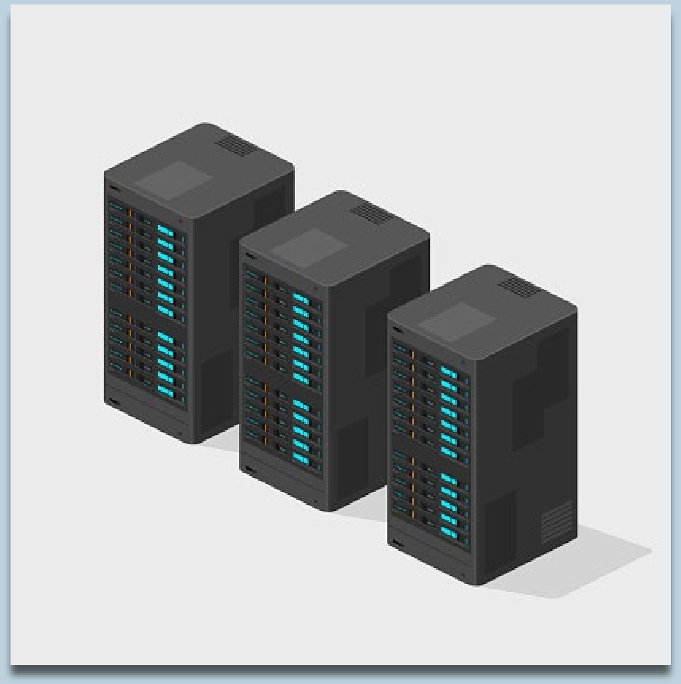
HOME | SELECT

DOWNLOAD OR REVISE EXTRACTS

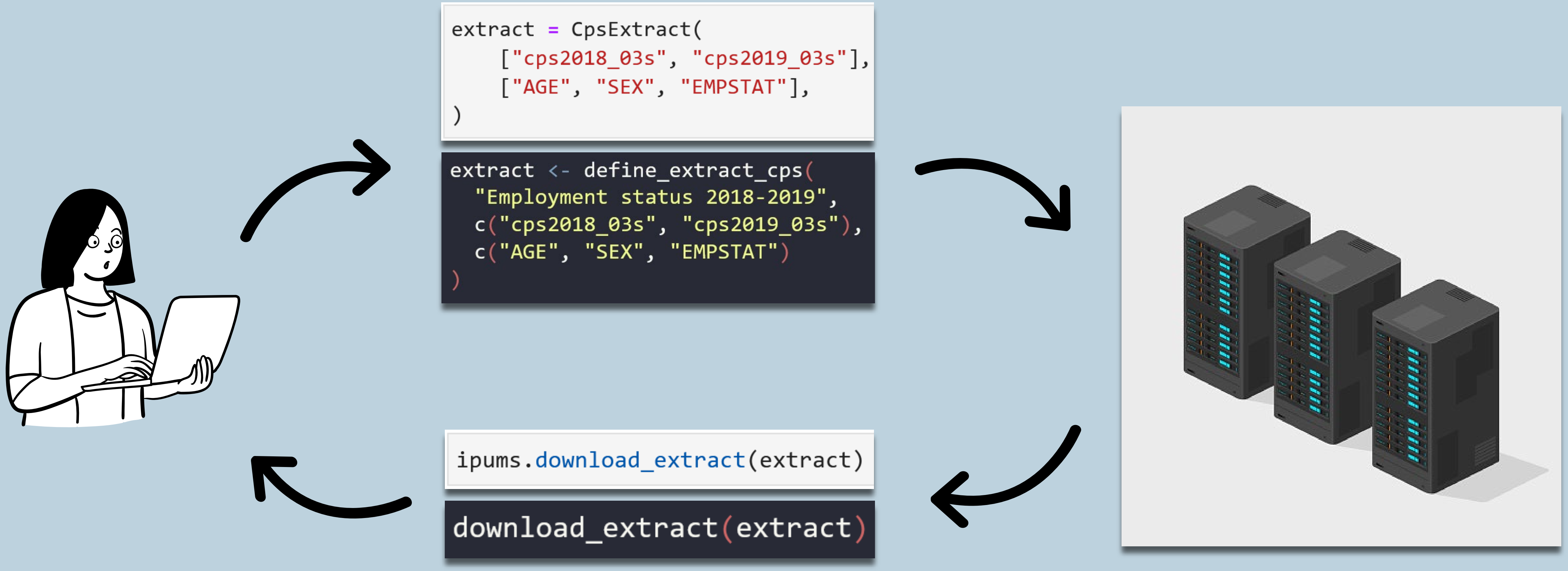
Use the links provided below to download a data extract (right-click the links for t
here. Note: data files will be available for 72 hours, after which they are subject to

« Previous | 1 | 2 | Next »

Extract		Formatted	Fixed-width Text Files		Codebook	R
Number	Date	Data	Data	Command Files		E
84	2022-07-12	--	Download .DAT	SPSS SAS Stata R	Basic DDI	
83	2022-06-22	--	--	- - - -		
82	2022-06-22	--	--	- - - -		
81	2022-06-09	--	--	- - - -		



IPUMS



```python
extract = CpsExtract(
    ["cps2018_03s", "cps2019_03s"],
    ["AGE", "SEX", "EMPSTAT"],
)
```

```r
extract <- define_extract_cps(
  "Employment status 2018-2019",
  c("cps2018_03s", "cps2019_03s"),
  c("AGE", "SEX", "EMPSTAT")
)
```

```python
ipums.download_extract(extract)
```

```r
download_extract(extract)
```

IPUMS

What can you do with the IPUMS API?

- Define and submit an extract
- Check extract status or “wait” for an extract to finish
- Download completed extracts
- Get info on past extracts
- Share extract definitions

What can’t you do with the IPUMS API?

- Bypass the extract system
- Explore what data are available
- Retrieve data from every IPUMS data collection
  - Only IPUMS CPS and IPUMS USA currently available
- Use all features of the online extract system (not yet)
  - But more are coming soon!

Why should I use the IPUMS API?

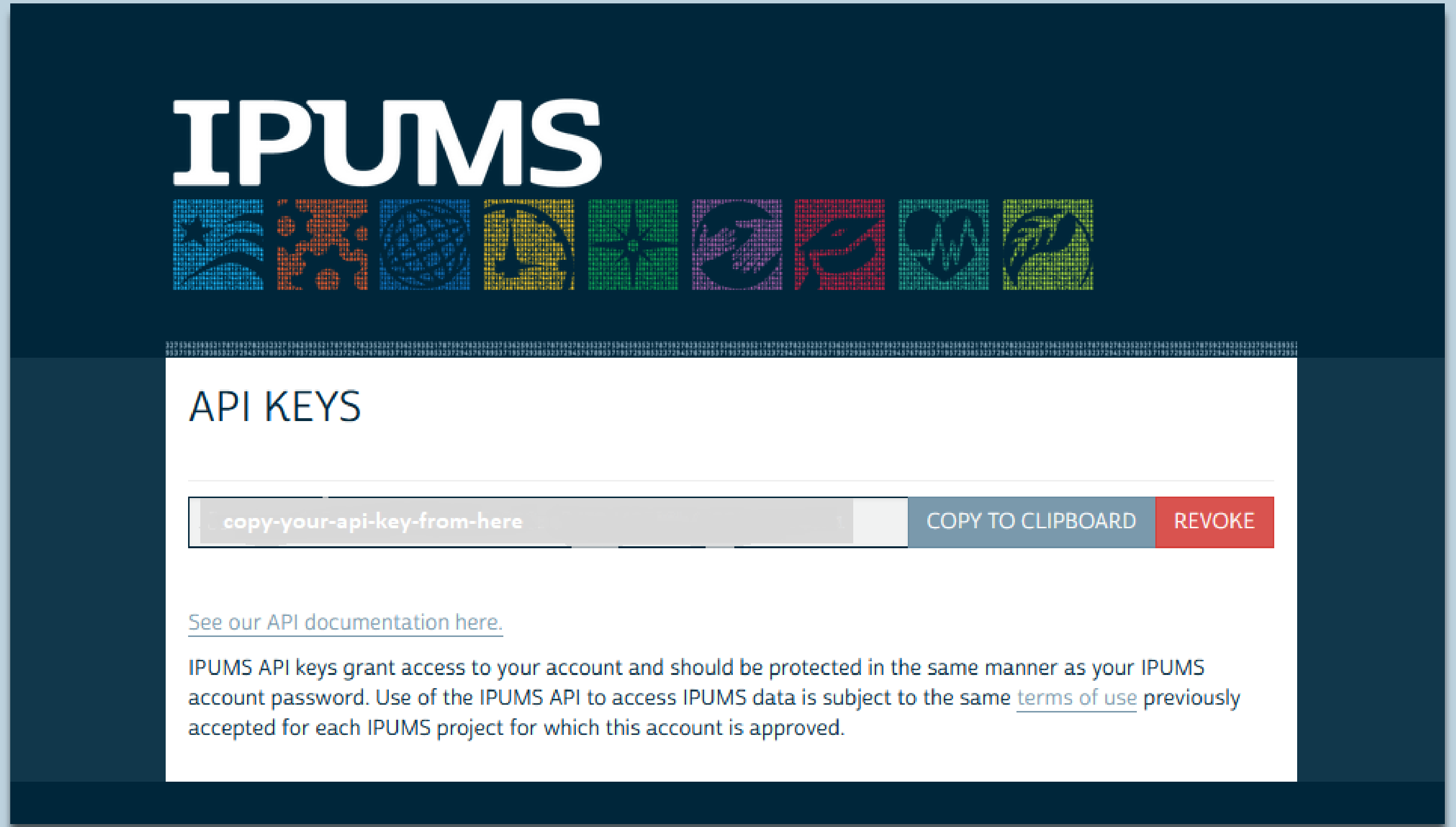
(HINT: think about sustainable workflows!)

1. Get IPUMS CPS data without leaving your code!
   - Easily update analyses with new samples or variables (So flexible!)
   - Caveat: Still a good idea to check the web documentation!

2. Share code or extract definitions and enhance reproducibility

Set Up: Getting your API key

https://account.ipums.org/api_keys



API KEYS

copy-your-api-key-from-here

See our API documentation here.

IPUMS API keys grant access to your account and should be protected in the same manner as your IPUMS account password. Use of the IPUMS API to access IPUMS data is subject to the same terms of use previously accepted for each IPUMS project for which this account is approved.

Hands-on with the API

IPUMS

Returning to our earlier analysis…

- Define an IPUMS CPS extract
- Submit that extract via the IPUMS API
- Wait for our extract to complete
- Download the data and DDI file
- Read that data into pandas

Defining Our Extract

- Here is the extract definition for the extract we used in our workflow discussion:

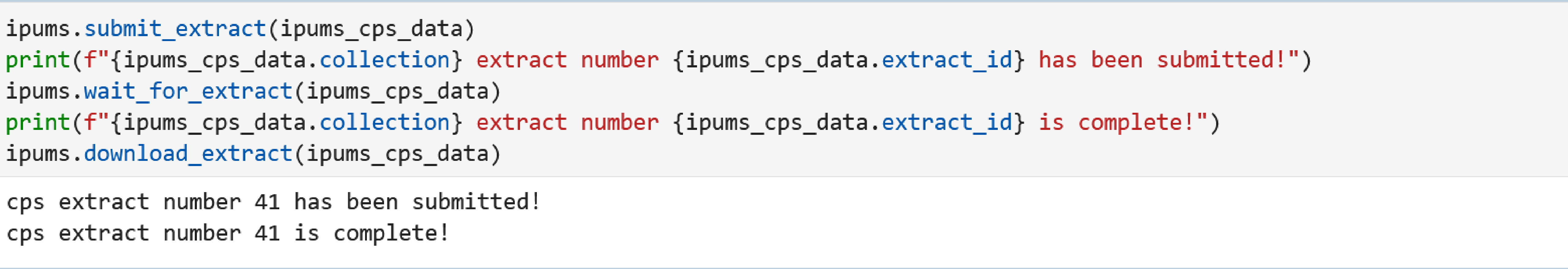
```
ipums_cps_data = CpsExtract(["cps2010_03s", "cps2011_03s", "cps2012_03s", "cps2013_03s", "cps2014_03s", "cps2015_03s",
                             "cps2016_03s", "cps2017_03s", "cps2018_03s", "cps2019_03s", "cps2020_03s", "cps2021_03s"],
                              ["SPMPOV", "SPMWT", "OFFPOV", "ASECWT"],
                              description="2010-2021 ASEC data, poverty variables")
```

Defining Our Extract

- Now that 2022 ASEC data is available, we can update this extract definition by simply updating the sample id list

```
ipums_cps_data = CpsExtract(["cps2010_03s", "cps2011_03s", "cps2012_03s", "cps2013_03s", "cps2014_03s", "cps2015_03s",
                             "cps2016_03s", "cps2017_03s", "cps2018_03s", "cps2019_03s", "cps2020_03s", "cps2021_03s",
                             "cps2022_03s"],
                              ["SPMPOV", "SPMWT", "OFFPOV", "ASECWT"],
                              description="2010-2021 ASEC data, poverty variables")
```

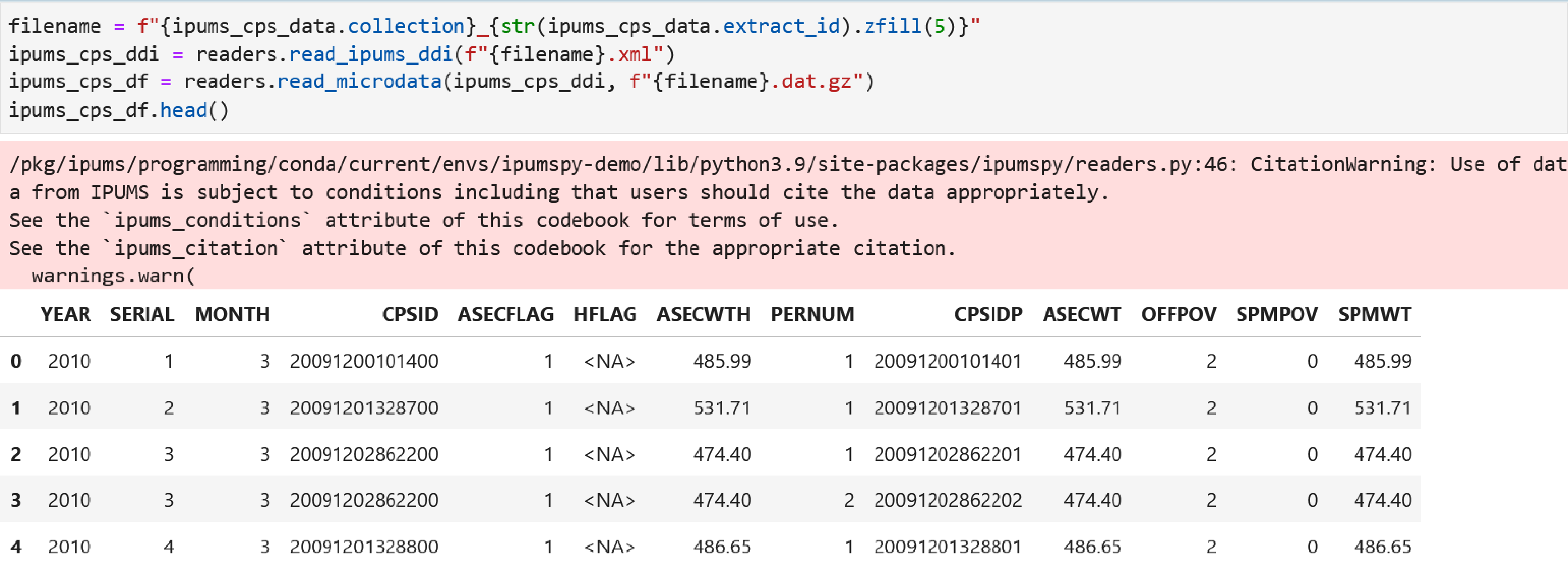
Submit, attend, and download



```python
ipums.submit_extract(ipums_cps_data)
print(f"{ipums_cps_data.collection} extract number {ipums_cps_data.extract_id} has been submitted!")
ipums.wait_for_extract(ipums_cps_data)
print(f"{ipums_cps_data.collection} extract number {ipums_cps_data.extract_id} is complete!")
ipums.download_extract(ipums_cps_data)
```

```
cps extract number 41 has been submitted!
cps extract number 41 is complete!
```

Read it in

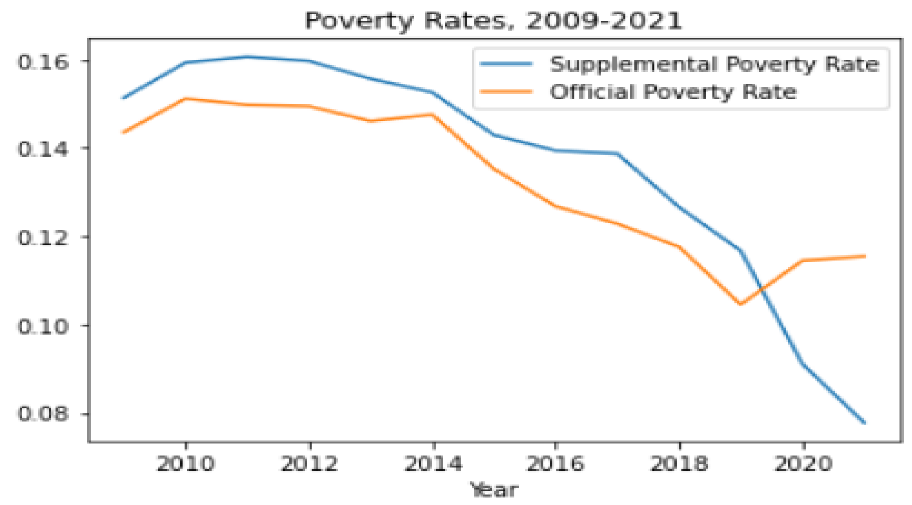


```python
filename = f"{ipums_cps_data.collection}_{str(ipums_cps_data.extract_id).zfill(5)}"
ipums_cps_ddi = readers.read_ipums_ddi(f"{filename}.xml")
ipums_cps_df = readers.read_microdata(ipums_cps_ddi, f"{filename}.dat.gz")
ipums_cps_df.head()
```

```
/pkg/ipums/programming/conda/current/envs/ipumspy-demo/lib/python3.9/site-packages/ipumspy/readers.py:46: CitationWarning: Use of dat
a from IPUMS is subject to conditions including that users should cite the data appropriately.
See the `ipums_conditions` attribute of this codebook for terms of use.
See the `ipums_citation` attribute of this codebook for the appropriate citation.
  warnings.warn(
```

	YEAR	SERIAL	MONTH	CPSID	ASECFLAG	HFLAG	ASECWTH	PERNUM	CPSIDP	ASECWT	OFFPOV	SPMPOV	SPMWT
0	2010	1	3	20091200101400	1	<NA>	485.99	1	20091200101401	485.99	2	0	485.99
1	2010	2	3	20091201328700	1	<NA>	531.71	1	20091201328701	531.71	2	0	531.71
2	2010	3	3	20091202862200	1	<NA>	474.40	1	20091202862201	474.40	2	0	474.40
3	2010	3	3	20091202862200	1	<NA>	474.40	2	20091202862202	474.40	2	0	474.40
4	2010	4	3	20091201328800	1	<NA>	486.65	1	20091201328801	486.65	2	0	486.65

```python
# distinguish between survey year and year
ipums_cps_df["SVY_YEAR"] = ipums_cps_df["YEAR"]
ipums_cps_df["YEAR"] = ipums_cps_df["SVY_YEAR"] - 1

# calculate spm poverty rates using our `get_pov_rate` function defined previously
spm_rt_by_yr = ipums_cps_df.groupby(["YEAR"]).apply(get_pov_rate,
                                                    pov_var="SPMPOV",
                                                    pov_wt="SPMWT",
                                                    in_pov_val=1)
# calculate official poverty rates using our `get_pov_rate` function defined previously
off_rt_by_yr = ipums_cps_df.groupby(["YEAR"]).apply(get_pov_rate,
                                                    pov_var="OFFPOV",
                                                    pov_wt="ASECWT",
                                                    in_pov_val=1,)

# graph it!
# plot spm rates first
spm_rt_by_yr.plot(label="Supplemental Poverty Rate")
# retrieve the current plot
ax = plt.gca()
# add official poverty rates
ax.plot(off_rt_by_yr, label="Official Poverty Rate")
# add a legend
ax.legend()
# specify plot title
ax.set_title(f"Poverty Rates, 2009-{max(ipums_cps_df['YEAR'])}")
# specify x axis label
ax.set_xlabel("Year")
plt.show()
```



Complete the rest of the analysis from the top

IPUMS

Further IPUMS API Resources

Using ipumsr to make an IPUMS extract

- ipumsr is an R package maintained by IPUMS for reading and manipulating IPUMS data and interacting with the IPUMS API
- ipumsr resources
  - Library documentation
  - API tutorial
  - Code snippets on IPUMS developer portal
  - Reproducible research with ipumsr – blog post

Using IPUMSPY to make an IPUMS extract

- ipumspy is a Python library maintained by IPUMS for working with IPUMS data and interacting with the IPUMS API
- ipumspy resources
  - Library documentation
  - Code snippets on IPUMS developer portal
  - Using ipumspy to share extract definitions – blog post

Using IPUMSPY to make an IPUMS extract in Stata

- Stata v 16+ supports using Python from within Stata
- Users can leverage ipumspy to make an extract from Stata
- Some one-time set up is required
  - [Making IPUMS Extracts from Stata]() – blog post

Questions?