# Web Scraping in Support of the U.S. Census Bureau's Public Sector Programs

**Hector Ferronato[1,2], Brian Dumbacher[1]**
Hector.R.Ferronato@census.gov, Brian.Dumbacher@census.gov
[1]U.S. Census Bureau, 4600 Silver Hill Road, Washington, DC 20233
[2]Reveal Global Consulting, 8115 Maple Lawn Blvd. #375, Fulton, MD 20759

Proceedings of the 2022 Federal Committee on Statistical Methodology (FCSM) Research and Policy Conference

## Abstract

The U.S. Census Bureau conducts many surveys of state and local governments to collect data on government employment and finances. For these surveys, respondent data or equivalent-quality data can sometimes be found on government websites. Analysts currently obtain data from these sources manually, so an automated approach would be very useful. A long-term web scraping solution must handle various online formats, structures, and content. To this end, the Census Bureau is developing methods that find documents from multiple URLs, convert the documents to TXT and EXCEL files, and perform large-scale scraping using predetermined key terms. This work is being performed using the SABLE (Scraping Assisted by Learning) web scraping environment, which is based on the open-source software Python. This report details current web scraping efforts in support of State Government Finances and the Quarterly Summary of State and Local Tax Revenue.

**Key Words:** U.S. Census Bureau, economic statistics, web scraping, government units

## 1. Introduction

### 1.1 Background

The Economic Directorate of the U.S. Census Bureau conducts various surveys of state and local governments. These surveys collect a wealth of data on government organization, employment and payroll, finances, and retirement systems (U.S. Census Bureau, 2022a). Respondent data or equivalent-quality data can often be found on government websites in the form of Comprehensive Annual Financial Reports and other publications. Going directly to government websites and collecting data passively has the potential to reduce burden for both respondents and Census Bureau analysts (Dumbacher and Hanna, 2017). Indeed, in many instances analysts obtain data from these online sources and use them for imputation and quality assurance. However, current passive data collection processes are manual. Using automated methods such as web scraping (Mitchell, 2015) and other data science techniques can greatly improve efficiency.

### 1.2 General Web Scraping Challenges

An automated process for scraping and organizing data is ideal but challenging to develop. For example, government websites and the documents on them reflect a wide variety of structures. This makes it difficult to find a solution that works consistently for all governments. Government publications also tend to be in Portable Document Format (PDF). This format ensures content is displayed the same regardless of operation system but does not lend itself well to immediate analysis. One solution is to convert PDFs to some intermediate format and then attempt to scrape. More direct data extraction methods exist, but they must still contend with data displayed in tables and images. Another challenge is that the location and format of the data may change over time. For example, governments may redesign their website or alter the layout of their publications.

### 1.3 Outline

The rest of the paper is organized as follows. Section 2 describes a computing environment called SABLE for conducting web scraping in support of the Census Bureau's public sector programs. Sections 3 and 4 detail two use cases that utilize SABLE. The first case involves scraping tax revenue data from various state government websites and publications for the Quarterly Summary of State and Local Tax Revenue. The second involves a deeper, more focused scraping effort in support of the State Government Finances program. Methodology and results for both use cases are discussed in detail. Lastly, Section 5 gives concluding remarks and lays out ideas for future work.

Disclaimer: Any opinions and conclusions expressed herein are those of the authors and do not reflect the views of the U.S. Census Bureau. The Census Bureau has reviewed this data product for unauthorized disclosure of confidential information and has approved the disclosure avoidance practices applied (Approval ID: CBDRB-FY22-ESMD001-013).

## 2. Scraping Assisted by Learning (SABLE)

The Census Bureau has created an environment for conducting web scraping in support of its public sector programs. This environment is called SABLE, which stands for Scraping Assisted by Learning. SABLE consists of two Linux servers and software for performing various web scraping tasks. Figure 1 illustrates the system architecture. SABLE received its Authority to Operate (ATO) in August 2018. The process of obtaining the ATO involved demonstrating that SABLE meets certain security requirements, documenting SABLE's administration and operation, and developing a change control process, among other things.

The main piece of software used by SABLE is Python, which is a popular programming language for data science applications. Python is used to download documents from government websites, scrape data from documents, and process the scraped data. There are six main Python modules: PDFMiner, Beautiful Soup, Tabula, Pandas, scikit-learn, and the Natural Language Toolkit (NLTK). PDFMiner (Shinyama, 2013), Beautiful Soup (Crummy, 2022), and Tabula (Ariga, 2019) offer solutions to extract and parse data from PDFs. The Pandas module (McKinney, 2010) provides many convenient data structures and methods for data manipulation. Scikit-learn is a commonly used machine learning module with many options for text classification (Pedregosa *et al.*, 2011). Lastly, NLTK provides many methods for analyzing text and building related models (Bird, 2006).

Another key piece of software installed on SABLE is Apache Nutch, which is a Java-based web crawler (Apache, 2022). The two use cases described in this paper rely on Python and its modules. For more information about Apache Nutch and SABLE's web crawling and machine learning capabilities, please see Dumbacher and Diamond (2018).

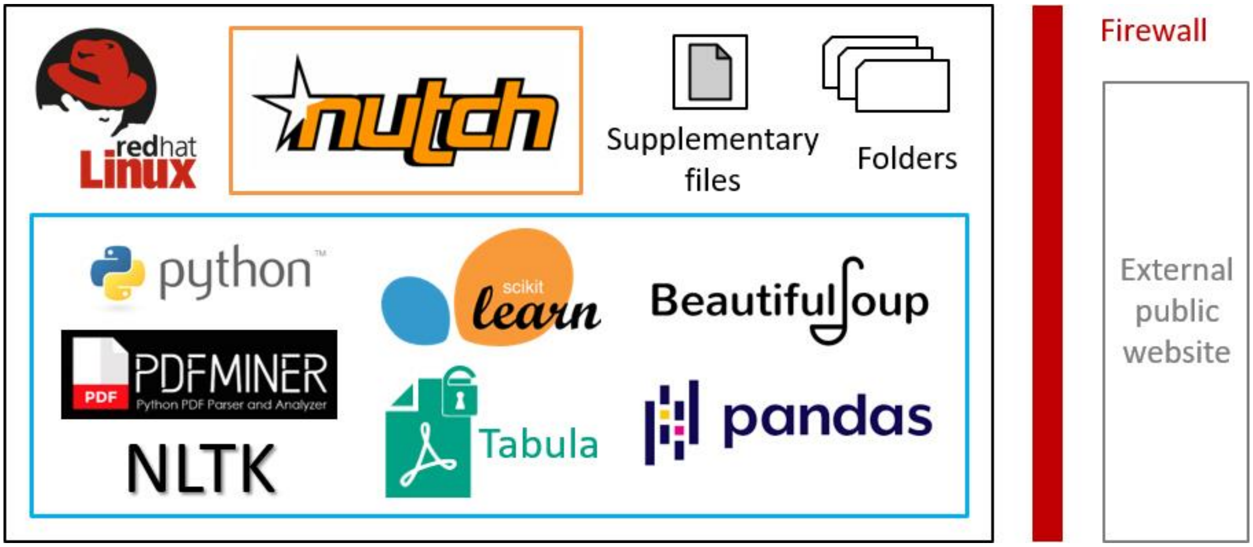


**Figure 1**. SABLE architecture design. SABLE resides on two Linux servers behind the Census Bureau's firewall and scrapes data from external public websites. Python and Apache Nutch are the two key pieces of software.

## 3. Quarterly Summary of State and Local Tax Revenue

The Quarterly Summary of State and Local Tax Revenue (QTAX) collects quarterly tax revenue data from state and local governments (U.S. Census Bureau, 2022b). The taxes in scope to QTAX include general sales and gross receipts tax, individual income tax, corporation net income tax, and property tax. For more information about QTAX, including technical documentation, see https://www.census.gov/programs-surveys/qtax.html. Much of this tax revenue data is publicly available on government websites in tax revenue collection reports and similar financial documents. Many of these online sources provide data on a more frequent, monthly basis. Collecting tax revenue data from government websites in an automated fashion could improve efficiency and the detail of the data collected. To this end, the Census Bureau is exploring the feasibility of collecting data for QTAX via web scraping. Current research is focused on scraping data for 26 of the 50 state governments and for all taxes of interest to the analysts. These 26 states were identified based on highest priority by Census Bureau analysts.

The methodology developed for this project involves a pair of Python programs. The first program iterates through the states and downloads specific documents from their websites. The second program then scrapes the desired tax revenue values from the downloaded documents. There is a pair of programs for each file format: PDF or EXCEL.

### 3.1 QTAX PDF Downloading

The Python program for downloading PDFs consists of the following steps:

- Take a user-specified year and month as input. These are the year and month of the desired tax revenue values (e.g., tax revenue collected by state governments in December 2019).
- Iterate through the states
- For each state, either:
    - Iterate through candidate Uniform Resource Locators (URLs) that are functions, or patterns, of the user-specified year and month
        - For each candidate URL
            - Attempt to download the PDF at the URL and convert it to TXT format
            - If either the download or TXT conversion is unsuccessful, then move on to the next URL pattern
        - If both the download and TXT conversion are successful, then move on to the next state
    - Or visit the state main archive website containing URLs to multiple PDF reports by date
        - For each month's PDF URL and date
            - Check whether the user-specified date matches the PDF URL date and attempt to download the PDF and convert it to TXT format

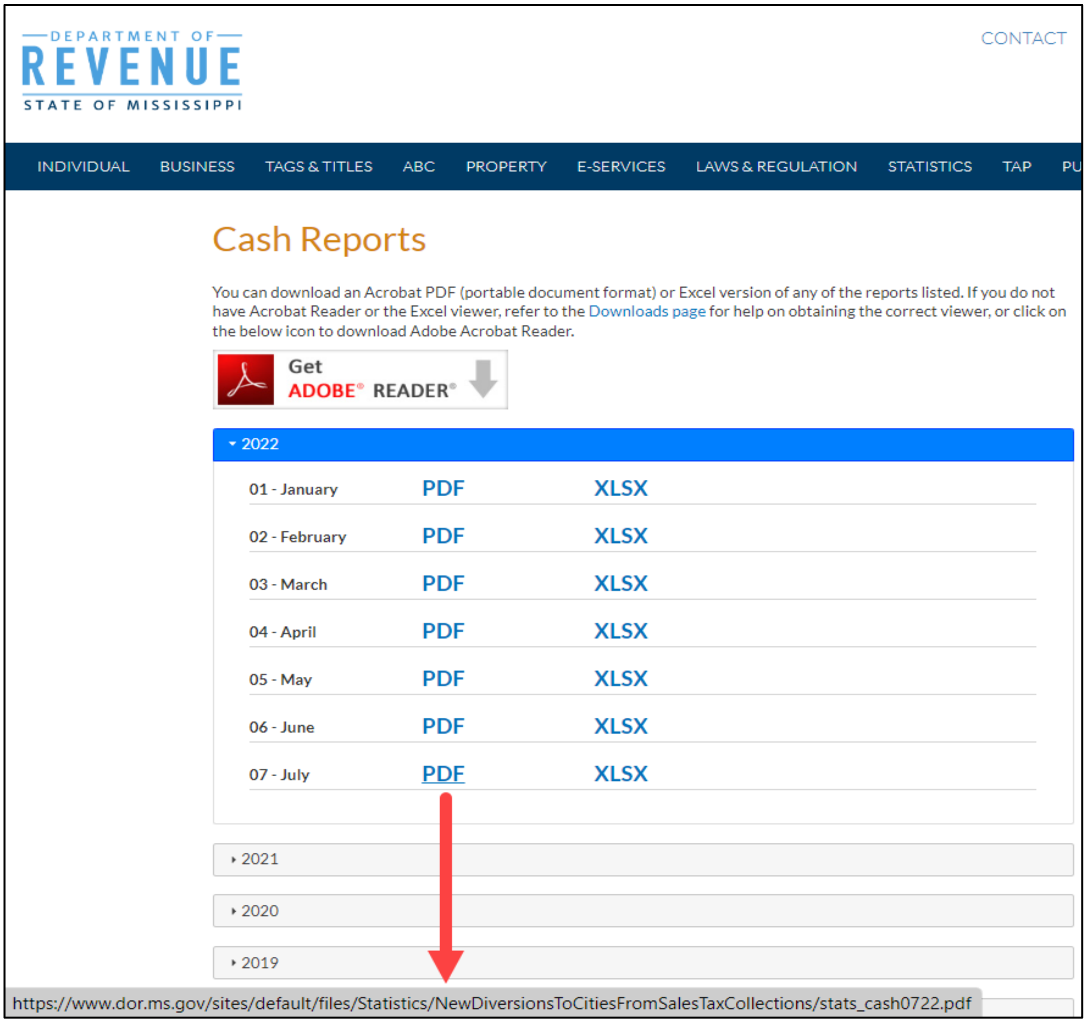
The analysts determine what PDFs to download for each state, and the researchers then propose candidate URLs that are functions of the user-specified year and month. These functions are referred to as URL patterns. URL patterns depend on the year and month in various ways, the most common of which are:

- Year and month (written out or expressed numerically)
- Fiscal year (depends on state) and month
- Year and month of document release
- Full date of document release

A state government's inconsistent use of the URL pattern from month to month means more patterns must be developed in order to generate more candidate URLs. Different patterns may be required because of inconsistent naming conventions, date formats, use of punctuation, and document storage locations. To illustrate this, below are two example URLs for the Vermont Agency of Administration's Monthly Revenue Release containing tax revenue data. The PDF location on the website varies, and the PDF name depends on year, fiscal year, month, and full date of document release. To deal with the full date of document release, there is a separate URL pattern for each possible date of the following month. The main challenge with this downloading method is URL inconsistency over time.

- https://aoa.vermont.gov/sites/aoa/files/revenue-economy/RevenueRpts/DECEMBER%20-%20FY20%20Revenue%20Press%20Release%201-24-20.pdf
- https://aoa.vermont.gov/sites/aoa/files/InfoReportReleases/Revenue%20Press%20Release%20FY20%20May%206-19-20.pdf

When available, analysts identify the main archive page that the state uses to publish the tax reports. In this case the Python program visits the archive and searches for the report URL. These archive pages contain tables and lists with links to the documents, accompanied by a report title and report date. By parsing the webpage HTML using Beautiful Soup, the Python program searches for elements such as tables, lists, and buttons and checks whether the table row or link corresponds to the report and date. If so, the URL is used to download the document. For an example of an archive page, see Figure 2. The code that performs this search is custom to each state tax report archive page because the layouts vary. For example Python code that performs this search for the Mississippi tax report archive page, see Code Block 1 in Appendix A.



Cash Reports

You can download an Acrobat PDF (portable document format) or Excel version of any of the reports listed. If you do not have Acrobat Reader or the Excel viewer, refer to the Downloads page for help on obtaining the correct viewer, or click on the below icon to download Adobe Acrobat Reader.

▾ 2022

| | | |
|---|---|---|
| 01 - January | PDF | XLSX |
| 02 - February | PDF | XLSX |
| 03 - March | PDF | XLSX |
| 04 - April | PDF | XLSX |
| 05 - May | PDF | XLSX |
| 06 - June | PDF | XLSX |
| 07 - July | PDF | XLSX |

▸ 2021

▸ 2020

▸ 2019

https://www.dor.ms.gov/sites/default/files/Statistics/NewDiversionsToCitiesFromSalesTaxCollections/stats_cash0722.pdf

**Figure 2.** Example of Mississippi tax reports archive webpage. The URL is identified by reading the date column in the same row as the report download button "PDF". Source: https://www.dor.ms.gov/statistics/cash-reports

In summary, the main difference between the two downloading methods is how the correct URL is found. With the pattern methodology, multiple URL patterns are generated based on an existing and validated URL. Variations are considered to increase the number of candidate URLs. The second downloading method consists of accessing a tax report archive page that does not vary monthly, parsing the webpage HTML code, and finding a single, correct URL within elements of the page (e.g., tables, lists, and buttons).

After finding the correct URL for the desired year and month, the Python program downloads the document using a Linux "wget" command. This command fetches the document using a certificate and saves the output in a specified location. The code below shows the details of "wget". A custom user-agent string includes URLs of websites that contain information about SABLE and the Census Bureau's web scraping policy.

```
os.system("wget --no-check-certificate -nv --user-agent=\"SABLE (U.S.
    Census Bureau research to find alternative data sources and reduce
    respondent burden) https://github.com/uscensusbureau/sable/; census-
    aidcrb-support-team@census.gov; For more information, go to
    www.census.gov/scraping/\" -P " + pdf_output_location)
```

Note that the document is downloaded with the original filename. The document is later given a standardized name that identifies the state, year, and month.

### 3.2 QTAX PDF Scraping

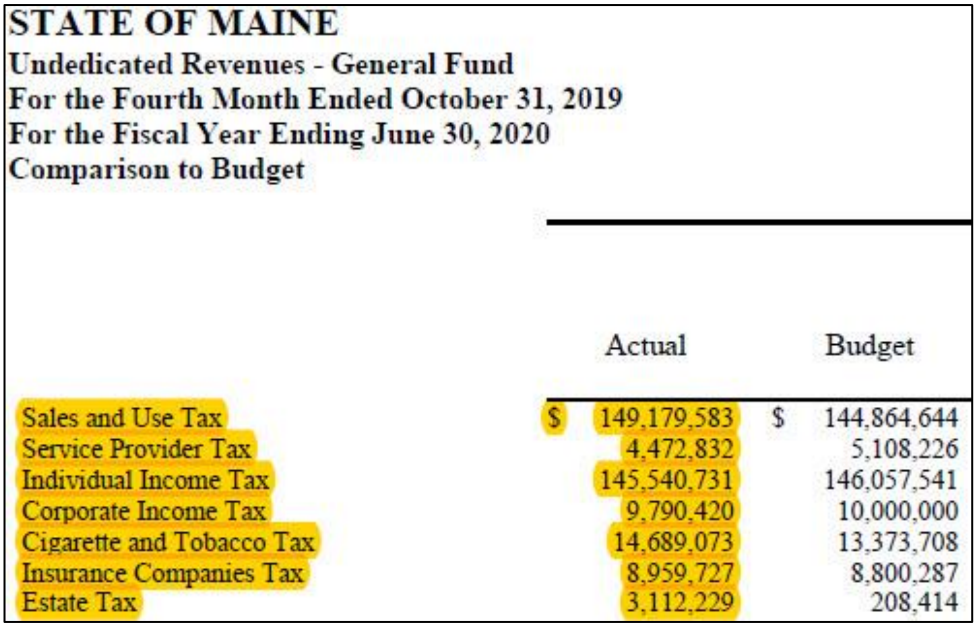
After the documents are downloaded, another Python program converts the PDFs to selectable TXT and searches for tax types and values. This program consists of the following detailed steps:

- Take a user-specified year and month as input. These are the year and month of the desired tax revenue values.
- Iterates through the states
- For each state
  - Apply a tailored scraping template to the downloaded PDF corresponding to the user-specified year and month
  - Extract the actual tax values and corresponding metadata (i.e., tax names, time period, and units)
- Organize and save the scraped data for all taxes and states in a single output TXT file

Analysts review the data sources and tell the researchers what taxes to scrape for each state. The researchers then develop a template that takes the TXT version of the PDF as input and scrapes the following data:

- Tax names
- Tax values
- Time period (e.g., monthly, or fiscal year to date)
- Units (e.g., dollars or thousands of dollars)

The scraping template consists of regular expressions (i.e., rules for matching patterns in text) and logic for dealing with match results. The Python program reads in the TXT version of the PDF line by line, applies the regular expressions, and scrapes accordingly. The template considers several factors: key words and phrases indicative of the location of the data (usually contained in tables); table structure; and tax value format. The template utilizes the user-specified year and month in certain cases to conduct scraping checks. Because the taxes of interest and key words and phrases vary from PDF to PDF, the template needs to be tailored. However, there is enough similarity in table structure and tax value format among PDFs that large portions of templates can be reused. As with downloading, a key challenge with scraping is inconsistency. The researchers endeavor to keep the regular expressions and logic general enough to accommodate minor variations in wording, table structure, and tax value format. Special treatment is needed for PDFs that exhibit major variations from month to month.



STATE OF MAINE
Undedicated Revenues - General Fund
For the Fourth Month Ended October 31, 2019
For the Fiscal Year Ending June 30, 2020
Comparison to Budget

| | | Actual | | Budget |
|---|---|---|---|---|
| Sales and Use Tax | $ | 149,179,583 | $ | 144,864,644 |
| Service Provider Tax | | 4,472,832 | | 5,108,226 |
| Individual Income Tax | | 145,540,731 | | 146,057,541 |
| Corporate Income Tax | | 9,790,420 | | 10,000,000 |
| Cigarette and Tobacco Tax | | 14,689,073 | | 13,373,708 |
| Insurance Companies Tax | | 8,959,727 | | 8,800,287 |
| Estate Tax | | 3,112,229 | | 208,414 |

**Figure 3.** Partial screenshot of Maine's October 2019 Monthly Revenue Report with taxes of interest highlighted.

As an example, Figure 3 shows a partial screenshot of Maine's October 2019 Monthly Revenue Report with the taxes of interest highlighted by the analysts. The tax values are found in a column named "Actual". "Actual" always seems to be the first column of the data table, but as a check, the scraping template looks at the order of the strings "Actual" and "Budget" and determines the correct column number. The following code accomplishes this.

```
m_col = re.search(r"(actual|budget)\s+(budget|actual)", line)
if m_col:
    if m_col.group(1) == "budget" and m_col.group(2) == "actual":
        col = 2
    elif m_col.group(1) == "actual" and m_col.group(2) == "budget":
        col = 1
```

As for the tax values themselves, the template applies regular expressions that account for variations such as inconsistent spacing between words and the possible use of dollar signs. The following line of code contains the regular expression for "estate tax". The parts highlighted in blue are used to identify the tax values in the first two columns. In this data table, the values are represented as unbroken strings of digits, commas, periods, and parentheses (parentheses are commonly used to express negative values).

```
m = re.search(r"estate\s+tax\s+\$?\s*([\d,.()]+)\s+\$?\s*([\d,.()]+)",
line)
```

At the end of the process, the scraped data are organized into both a TXT file and EXCEL file to facilitate the analysts' validation afterwards.

There are also a couple limitations regarding this scraping method. The PDF-to-TXT conversion process does not convert images to text. Therefore, data contained in images cannot be extracted. In this case, a method such as Optical Character Recognition (OCR), which is beyond the scope of SABLE, would have to be used. The PDF-to-TXT conversion process does a very good job maintaining document structure, but it is not perfect. Partial loss of text and strange spacing issues occur rarely.

### 3.3 QTAX EXCEL Downloading

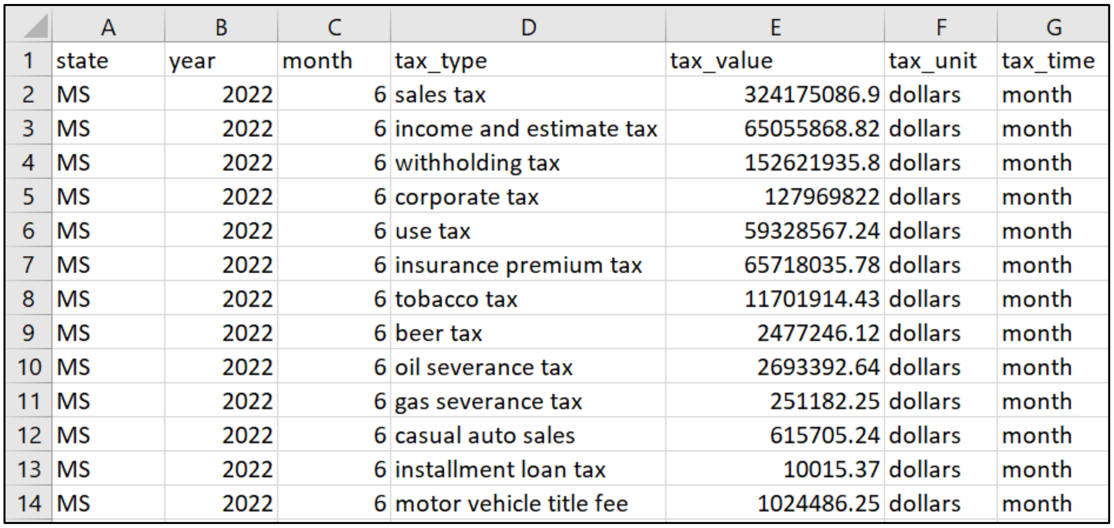
The process for downloading EXCEL files is very similar to the process for downloading PDFs described in Section 3.1. After the downloading program finds candidate URLs via patterns or via an archive page search, the files are automatically saved as EXCEL due to the extension in the file name (".xlsx"). The files are moved from temporary downloading directory to the correct folder for EXCEL files in the project structure.

### 3.4 QTAX EXCEL Scraping

Similar to PDF scraping, the EXCEL scraping program also searches for pre-defined tax types and corresponding values in the downloaded document. The EXCEL reports are not converted to TXT. Instead, the Pandas Python module reads the EXCEL report row by row, accesses it column by column, and collects tax values corresponding to the tax types that match the pre-defined tax type search terms. Each state EXCEL report requires customized code in order to use the correct tab and start reading the sheet at the correct header row. For example Python code that scrapes EXCEL files for Nevada, see Code Block 2 in Appendix A. The tax types and values scraped from the EXCEL report are aggregated and exported to a final EXCEL file containing only tax types and values specific to the specified state, year, and month. Figure 4 shows an example of the final EXCEL file.

### 3.5 Multi-Page Reports

Some states publish data in multiple reports, which analysts would like to scrape together into a single final output file. This requires a mix of downloading and scraping both PDF and EXCEL sources and then merging the results into the final scraped data.



|  | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| 1 | state | year | month | tax_type | tax_value | tax_unit | tax_time |
| 2 | MS | 2022 | 6 | sales tax | 324175086.9 | dollars | month |
| 3 | MS | 2022 | 6 | income and estimate tax | 65055868.82 | dollars | month |
| 4 | MS | 2022 | 6 | withholding tax | 152621935.8 | dollars | month |
| 5 | MS | 2022 | 6 | corporate tax | 127969822 | dollars | month |
| 6 | MS | 2022 | 6 | use tax | 59328567.24 | dollars | month |
| 7 | MS | 2022 | 6 | insurance premium tax | 65718035.78 | dollars | month |
| 8 | MS | 2022 | 6 | tobacco tax | 11701914.43 | dollars | month |
| 9 | MS | 2022 | 6 | beer tax | 2477246.12 | dollars | month |
| 10 | MS | 2022 | 6 | oil severance tax | 2693392.64 | dollars | month |
| 11 | MS | 2022 | 6 | gas severance tax | 251182.25 | dollars | month |
| 12 | MS | 2022 | 6 | casual auto sales | 615705.24 | dollars | month |
| 13 | MS | 2022 | 6 | installment loan tax | 10015.37 | dollars | month |
| 14 | MS | 2022 | 6 | motor vehicle title fee | 1024486.25 | dollars | month |

**Figure 4.** Screenshot of final output of scraped tax data for Mississippi from June 2022.

### 3.6 QTAX Results

Data have been collected for a total of 26 states for six months – January 2022 through June 2022. The resulting scraped files were validated by the analysts, and the states were classified into three categories according to success in downloading and scraping in an automated fashion.

States success categories in downloading:

- Good
  - Consistency in URL patterns
  - Consistency in URLs archive page
  - Regularly released documents
- Caution
  - Some inconsistency in URL patterns
  - Some inconsistency in URL archive page
  - Irregularly released documents
  - Some difficulty connecting to website server
- Problem
  - Documents not in PDF or EXCEL format
  - No monthly/quarterly data sources

Similarly, the states are classified into three categories according to success in scraping:

- Good
  - Consistency in document layout
  - No PDF-to-TXT conversion problems
  - No EXCEL scraping problems
- Caution
  - Some inconsistency in document layout
  - Some PDF-to-TXT conversion problems
  - Some EXCEL scraping problems
- Problem
  - Inconsistency in document layout
  - PDF-to-TXT conversion problems
  - No PDFs from which to scrape
  - EXCEL scraping problems
  - No EXCEL from which to scrape

**Table 1.** Classification of results from 26 states in terms of success in downloading and success in scraping

| Month | Downloading | | | Scraping | | |
|---|---|---|---|---|---|---|
| | Success | Caution | Problem | Success | Caution | Problem |
| Jan 2022 | 23 | 2 | 1 | 22 | 1 | 0 |
| Feb 2022 | 23 | 2 | 1 | 23 | 0 | 0 |
| Mar 2022 | 23 | 2 | 1 | 23 | 0 | 0 |
| Apr 2022 | 23 | 2 | 1 | 23 | 0 | 0 |
| May 2022 | 24 | 2 | 0 | 24 | 0 | 0 |
| Jun 2022 | 24 | 2 | 0 | 24 | 0 | 0 |

| | | | | | | |
|---|---|---|---|---|---|---|
| Total | 140 | 12 | 4 | 139 | 1 | 0 |
| Efficacy % | **90%** | 8% | 3% | **89%** | 1% | 0% |
| | 139 scraped out of 140 downloaded = **99.3% scraping efficacy** | | | | | |

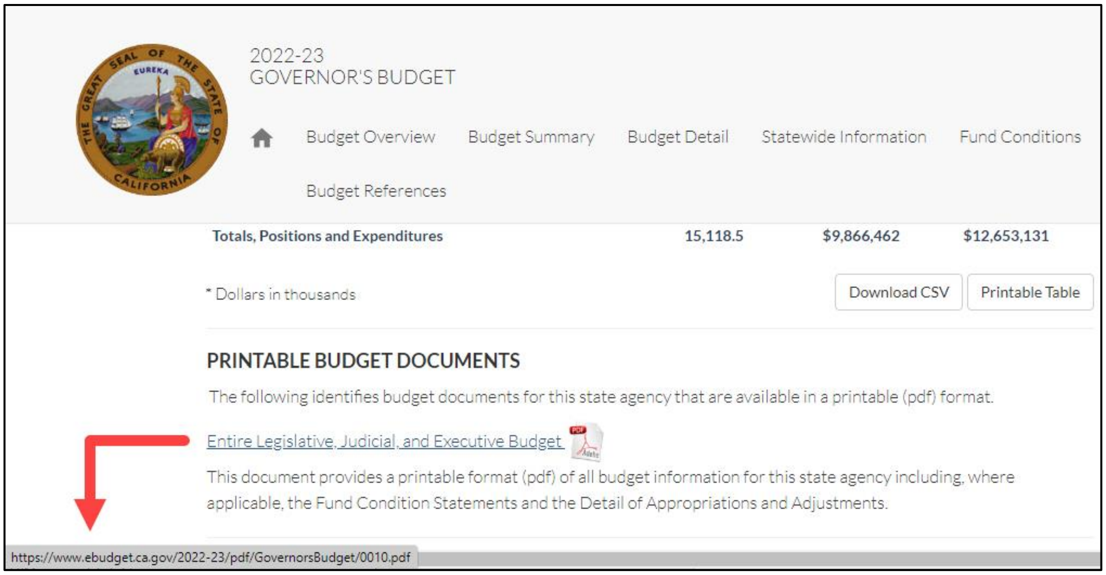
Table 1 summarizes results from the 26 state governments under consideration. The total is the column sum, and the efficacy percent is calculated as this value divided by 156 (= 6 × 26) combinations of month and state. The results show 90% efficacy for downloading and 89% for scraping across six months. When considering the scraping success rate from the total of 140 downloaded reports, the scraping efficacy increases to 99.3%. Some of the issues found were inconsistency or lack of frequency from the states on providing the reports for download, as well as minor layout changes in the reports themselves. For instance, Missouri and South Carolina only publish reports for download related to the current month of the fiscal year, and New York is inconsistent with its publishing dates. Also, occasionally new tax types are needed for collection. When this occurs, analysts need to update labeled files (e.g., Figure 3) so the scraping programs can be refined. Overall, the results show it is possible to automate the downloading and scraping of tax revenues from multiple states and reports across time.

## 4. State Government Finances

The State Government Finances (STATEFIN) program collects a wealth of data on government finances (U.S. Census Bureau, 2022c). The statistics released by STATEFIN include revenue by source, expenditures by function, debt by term, and assets by purpose. For more information about STATEFIN, including technical documentation, see https://www.census.gov/programs-surveys/state.html. Some states provide this data in PDFs published on their websites on a yearly basis. Automated data collection from governmental websites for state finances might increase the accuracy and detail of the data obtained as well as efficiency of the process. The Census Bureau is investigating the viability of using web scraping to collect data for STATEFIN, starting with a prototype focused on two California state finance reports: the Legislative, Judicial, and Executive section of the Governor's Budget (referred to as the 0010 report) and the Business, Consumer Services, and Housing Agency (1000).

### 4.1 STATEFIN PDF Downloading

California state government publishes its state finance reports yearly on a standardized Ebudget website, which displays the Governor's budget in a categorized and detailed manner (https://www.ebudget.ca.gov/budget/2022-23/#/BudgetDetail). See Figure 5 for a screenshot of this website. For each of the State Agencies in the Budget Detail section, there is another webpage with a PDF report of the entire agency's budget available for download. Currently, analysts manually download these reports. The goal of this prototype is to download the reports in an automated fashion similar to what was done for QTAX. This process involves finding the URLs for the reports based on the HTML webpage elements such as links and tables.



| Totals, Positions and Expenditures | 15,118.5 | $9,866,462 | $12,653,131 |
|---|---|---|---|

* Dollars in thousands

PRINTABLE BUDGET DOCUMENTS

The following identifies budget documents for this state agency that are available in a printable (pdf) format.

Entire Legislative, Judicial, and Executive Budget

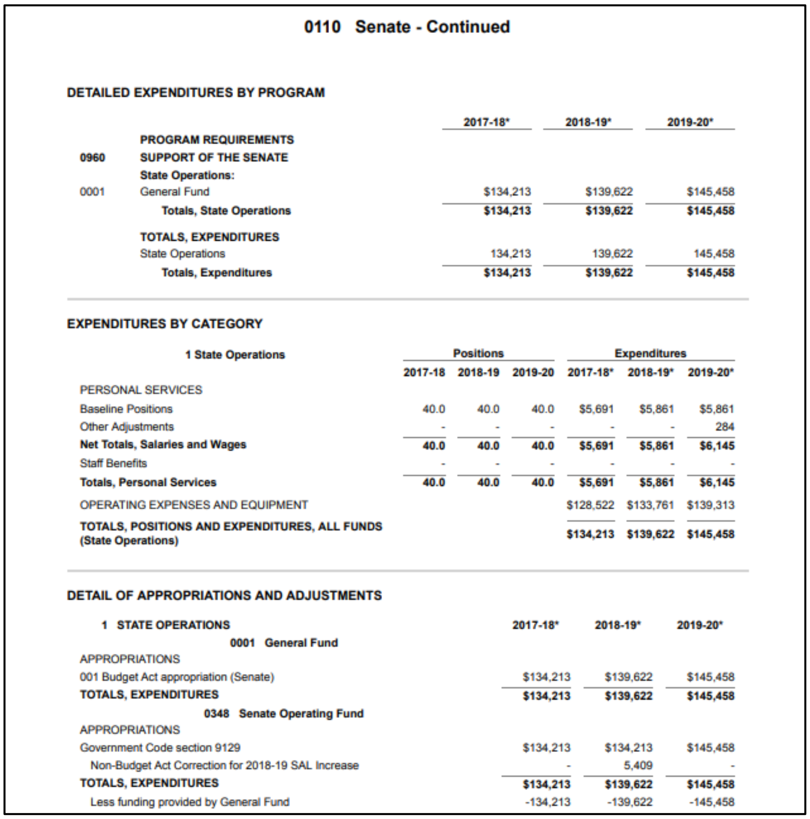
This document provides a printable format (pdf) of all budget information for this state agency including, where applicable, the Fund Condition Statements and the Detail of Appropriations and Adjustments.

https://www.ebudget.ca.gov/2022-23/pdf/GovernorsBudget/0010.pdf

**Figure 5.** Screenshot of California's Ebudget website with PDF report URL. Source: https://www.ebudget.ca.gov/budget/2022-23/#/BudgetDetail

### 4.2 STATEFIN PDF-to-EXCEL Conversion

After downloading, each report needs to be converted to EXCEL for labeling and for scraping. All PDF reports on California's Ebudget website have similar layouts, but they mix text descriptions and numerical tables. The position and length of the tables also vary by agency. To illustrate this, Figure 6 shows a screenshot of the 0010 PDF.



**0110 Senate - Continued**

DETAILED EXPENDITURES BY PROGRAM

| | | 2017-18* | 2018-19* | 2019-20* |
|---|---|---|---|---|
| | PROGRAM REQUIREMENTS | | | |
| 0960 | SUPPORT OF THE SENATE | | | |
| | State Operations: | | | |
| 0001 | General Fund | $134,213 | $139,622 | $145,458 |
| | Totals, State Operations | $134,213 | $139,622 | $145,458 |
| | TOTALS, EXPENDITURES | | | |
| | State Operations | 134,213 | 139,622 | 145,458 |
| | Totals, Expenditures | $134,213 | $139,622 | $145,458 |

EXPENDITURES BY CATEGORY

| 1 State Operations | Positions | | | Expenditures | | |
|---|---|---|---|---|---|---|
| | 2017-18 | 2018-19 | 2019-20 | 2017-18* | 2018-19* | 2019-20* |
| PERSONAL SERVICES | | | | | | |
| Baseline Positions | 40.0 | 40.0 | 40.0 | $5,691 | $5,861 | $5,861 |
| Other Adjustments | - | - | - | - | - | 284 |
| Net Totals, Salaries and Wages | 40.0 | 40.0 | 40.0 | $5,691 | $5,861 | $6,145 |
| Staff Benefits | - | - | - | - | - | - |
| Totals, Personal Services | 40.0 | 40.0 | 40.0 | $5,691 | $5,861 | $6,145 |
| OPERATING EXPENSES AND EQUIPMENT | | | | $128,522 | $133,761 | $139,313 |
| TOTALS, POSITIONS AND EXPENDITURES, ALL FUNDS (State Operations) | | | | $134,213 | $139,622 | $145,458 |

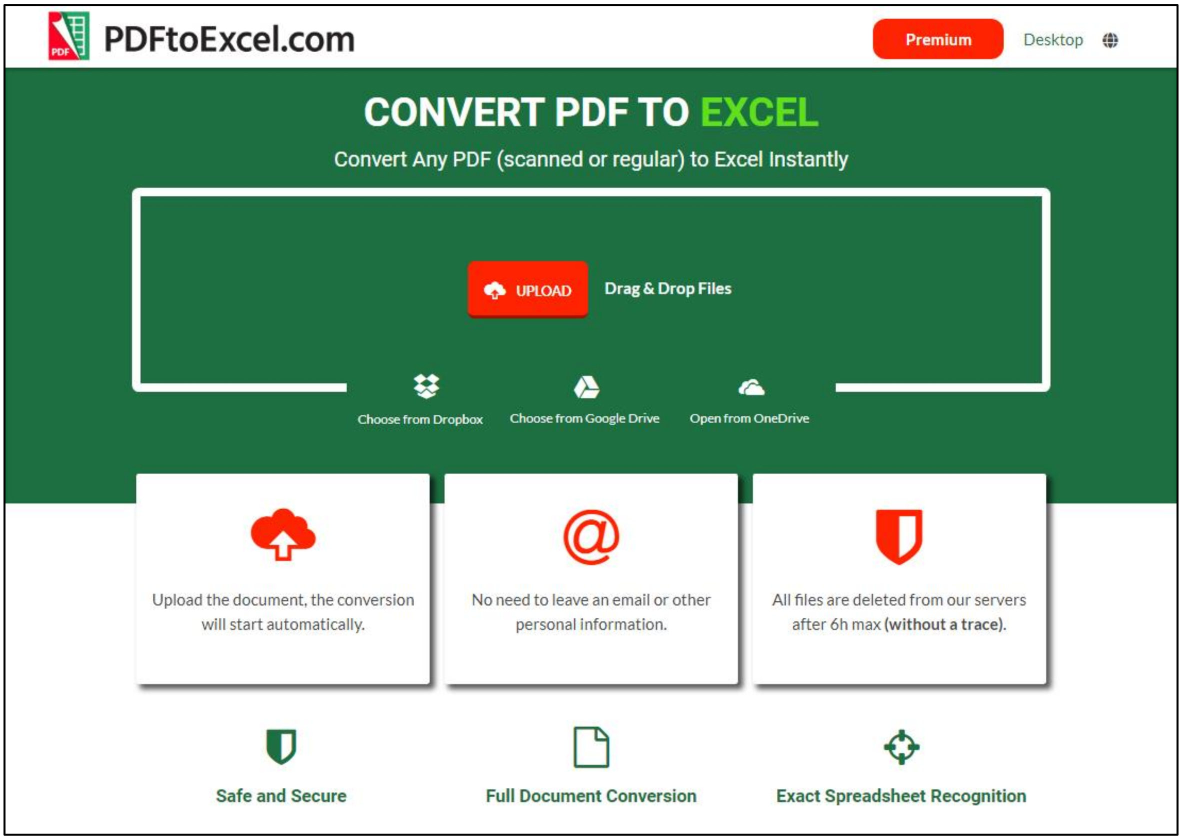
DETAIL OF APPROPRIATIONS AND ADJUSTMENTS

| 1 STATE OPERATIONS | 2017-18* | 2018-19* | 2019-20* |
|---|---|---|---|
| 0001 General Fund | | | |
| APPROPRIATIONS | | | |
| 001 Budget Act appropriation (Senate) | $134,213 | $139,622 | $145,458 |
| TOTALS, EXPENDITURES | $134,213 | $139,622 | $145,458 |
| 0348 Senate Operating Fund | | | |
| APPROPRIATIONS | | | |
| Government Code section 9129 | $134,213 | $134,213 | $145,458 |
| Non-Budget Act Correction for 2018-19 SAL Increase | - | 5,409 | - |
| TOTALS, EXPENDITURES | $134,213 | $139,622 | $145,458 |
| Less funding provided by General Fund | -134,213 | -139,622 | -145,458 |

**Figure 6.** Screenshot of PDF report 0010 downloaded from the California state government Ebudget website. Source: https://www.ebudget.ca.gov/2019-20/pdf/GovernorsBudget/0010.pdf

Converting these reports to EXCEL is challenging because there is no standard pattern or position for the tables containing the state finance objects and values. To overcome this, researchers are testing multiple tools to determine which conversion process best facilitates scraping. So far, two different methods have been tested for converting PDFs to EXCEL. The comparison of these methods is found below:

- PDFtoExcel.com, a secure website that offers PDF-to-EXCEL conversion
  - This method consists of uploading an original PDF to an HTTPS-secured website that automatically converts the PDF to EXCEL format and makes it available for download. Figure 7 shows a screenshot of PDFtoExcel.com.



CONVERT PDF TO EXCEL

Convert Any PDF (scanned or regular) to Excel Instantly

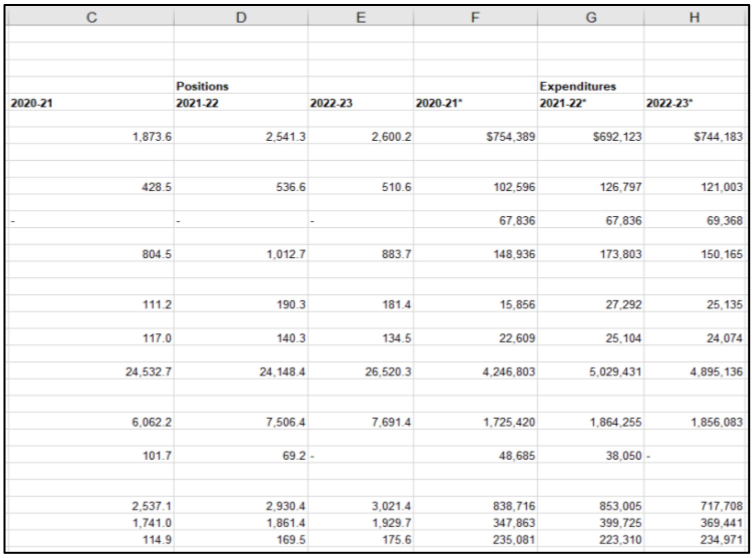
Upload the document, the conversion will start automatically.

No need to leave an email or other personal information.

All files are deleted from our servers after 6h max **(without a trace)**.

**Figure 7**. Screenshot of PDFtoExcel.com, a website that can be used to convert PDF reports to EXCEL files. Source: https://www.PDFtoExcel.com

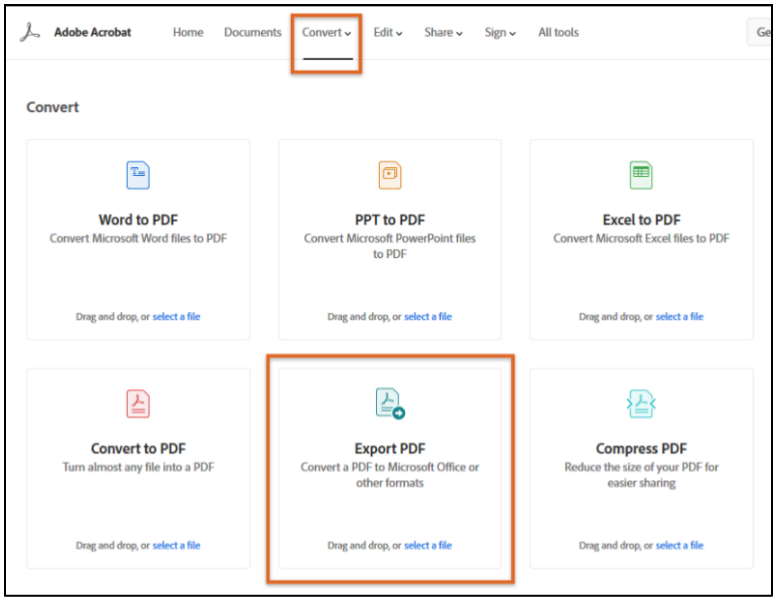
  - The benefit of this method is that it creates fewer columns in the final converted EXCEL compared to the other methods. The data then become well aligned. In turn, this makes it possible to ingest the data into a Pandas DataFrame, read it row by row, and find precisely where the tables and values are. For an example of the converted output, see Figure 8. The downside is the risk associated with accessing the website, although HTTPS secured, and downloading the converted EXCEL file.



| C | D | E | F | G | H |
|---|---|---|---|---|---|
| | Positions | | | Expenditures | |
| 2020-21 | 2021-22 | 2022-23 | 2020-21* | 2021-22* | 2022-23* |
| 1,873.6 | 2,541.3 | 2,600.2 | $754,389 | $692,123 | $744,183 |
| 428.5 | 536.6 | 510.6 | 102,596 | 126,797 | 121,003 |
| - | - | - | 67,836 | 67,836 | 69,368 |
| 804.5 | 1,012.7 | 883.7 | 148,936 | 173,803 | 150,165 |
| 111.2 | 190.3 | 181.4 | 15,856 | 27,292 | 25,135 |
| 117.0 | 140.3 | 134.5 | 22,609 | 25,104 | 24,074 |
| 24,532.7 | 24,148.4 | 26,520.3 | 4,246,803 | 5,029,431 | 4,895,136 |
| 6,062.2 | 7,506.4 | 7,691.4 | 1,725,420 | 1,864,255 | 1,856,083 |
| 101.7 | 69.2 | - | 48,685 | 38,050 | - |
| 2,537.1 | 2,930.4 | 3,021.4 | 838,716 | 853,005 | 717,708 |
| 1,741.0 | 1,861.4 | 1,929.7 | 347,863 | 399,725 | 369,441 |
| 114.9 | 169.5 | 175.6 | 235,081 | 223,310 | 234,971 |

**Figure 8.** Screenshot of the resulting EXCEL file after conversion using PDFtoExcel.com. It contains fewer and better aligned columns.

- Adobe Acrobat Reader
  - The Adobe Acrobat Reader local desktop program can convert PDFs to EXCEL format. It requires an Adobe registered account for online access or an extended tools package for the local desktop installation. Figure 9 shows a screenshot of the Adobe program.



Convert

**Word to PDF**
Convert Microsoft Word files to PDF
Drag and drop, or select a file

**PPT to PDF**
Convert Microsoft PowerPoint files to PDF
Drag and drop, or select a file

**Excel to PDF**
Convert Microsoft Excel files to PDF
Drag and drop, or select a file

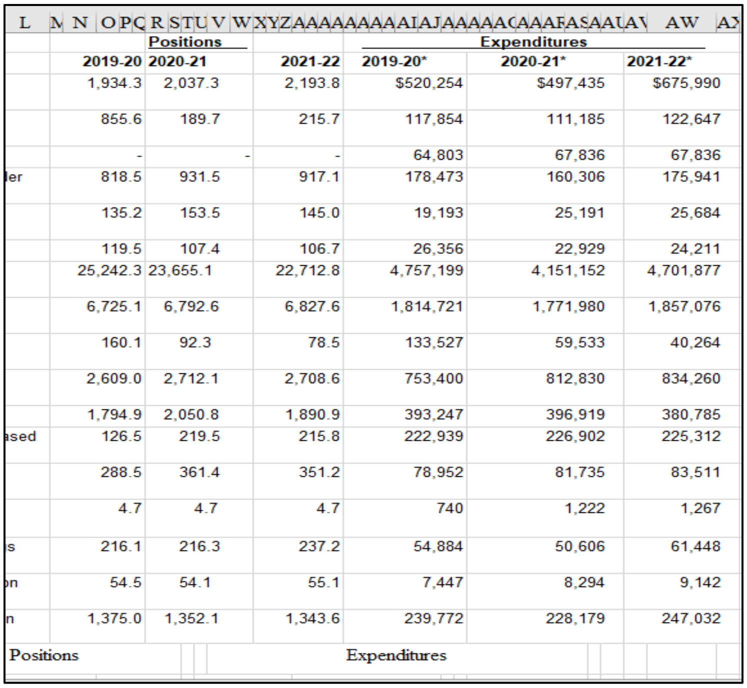
**Convert to PDF**
Turn almost any file into a PDF
Drag and drop, or select a file

**Export PDF**
Convert a PDF to Microsoft Office or other formats
Drag and drop, or select a file

**Compress PDF**
Reduce the size of your PDF for easier sharing
Drag and drop, or select a file

**Figure 9**. Screenshot of Adobe method for exporting PDF to EXCEL.

  - This method offers more security in terms of using a conversion method from the same source as the PDF (Adobe). However, as displayed in Figure 10, the conversion results in an excessive number of columns, many of which can be merged into longer cells to represent the titles of tables. The scraping programs do not read these resulting files well because the columns are too different row by row, and the merged cells makes it even more difficult to locate the tables.



L M N O P Q R S T U V W X Y Z A A A A A A A A A I A J A A A A A C A A A F A S A A U A V AW AX

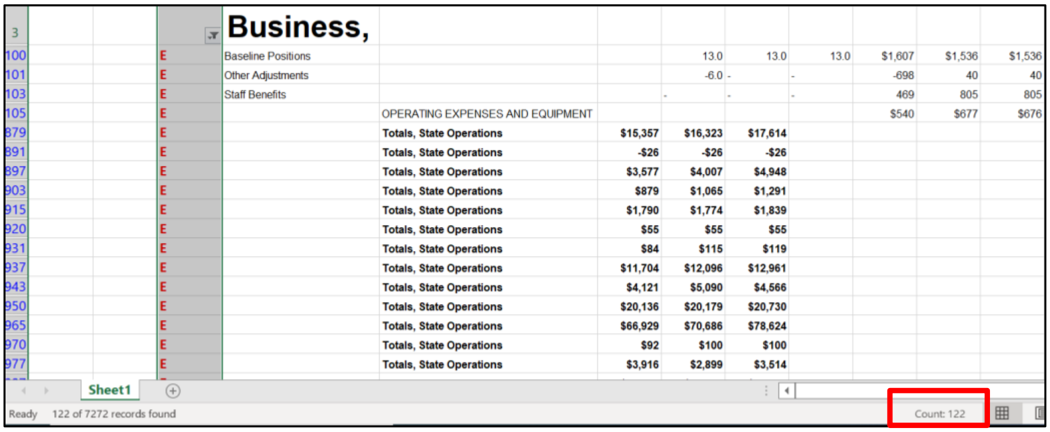
| | Positions | | | Expenditures | | |
|---|---|---|---|---|---|---|
| | 2019-20 | 2020-21 | 2021-22 | 2019-20* | 2020-21* | 2021-22* |
| | 1,934.3 | 2,037.3 | 2,193.8 | $520,254 | $497,435 | $675,990 |
| | 855.6 | 189.7 | 215.7 | 117,854 | 111,185 | 122,647 |
| | - | - | - | 64,803 | 67,836 | 67,836 |
| ler | 818.5 | 931.5 | 917.1 | 178,473 | 160,306 | 175,941 |
| | 135.2 | 153.5 | 145.0 | 19,193 | 25,191 | 25,684 |
| | 119.5 | 107.4 | 106.7 | 26,356 | 22,929 | 24,211 |
| | 25,242.3 | 23,655.1 | 22,712.8 | 4,757,199 | 4,151,152 | 4,701,877 |
| | 6,725.1 | 6,792.6 | 6,827.6 | 1,814,721 | 1,771,980 | 1,857,076 |
| | 160.1 | 92.3 | 78.5 | 133,527 | 59,533 | 40,264 |
| | 2,609.0 | 2,712.1 | 2,708.6 | 753,400 | 812,830 | 834,260 |
| | 1,794.9 | 2,050.8 | 1,890.9 | 393,247 | 396,919 | 380,785 |
| ased | 126.5 | 219.5 | 215.8 | 222,939 | 226,902 | 225,312 |
| | 288.5 | 361.4 | 351.2 | 78,952 | 81,735 | 83,511 |
| | 4.7 | 4.7 | 4.7 | 740 | 1,222 | 1,267 |
| s | 216.1 | 216.3 | 237.2 | 54,884 | 50,606 | 61,448 |
| on | 54.5 | 54.1 | 55.1 | 7,447 | 8,294 | 9,142 |
| n | 1,375.0 | 1,352.1 | 1,343.6 | 239,772 | 228,179 | 247,032 |
| Positions | | | | Expenditures | | |

**Figure 10.** Screenshot of the resulting EXCEL file after conversion using Adobe. The conversion yields too many columns, which are sometimes merged into longer cells.

Research shows that PDFtoExcel.com performs better due to the minimal number of columns in the final EXCEL file. However, guidance is being sought from the Census Bureau's information technology security and policy areas regarding use of the website. Future tasks involve trying other methods such as the Tabula Python module to perform PDF-to-EXCEL conversion.

## 4.3 STATEFIN Labeling

Census Bureau analysts receive the EXCEL version of each agency's report and label only the rows containing either a department, a program, or an object – the specific expense name followed by the dollar value. This is a significant difference between the labels from QTAX, which are only one tax type for each tax value. For STATEFIN, the label for each expenditure value is a combination of department, program, and object. These labels are essential for scraping, as the code will check only these rows while scraping data and formatting the final EXCEL file. In summary, the labels guide the automated scraping program to the useful information among tens of thousands of rows. For report 1000, analysts labeled 122 objects; see Figure 11 for a screenshot.



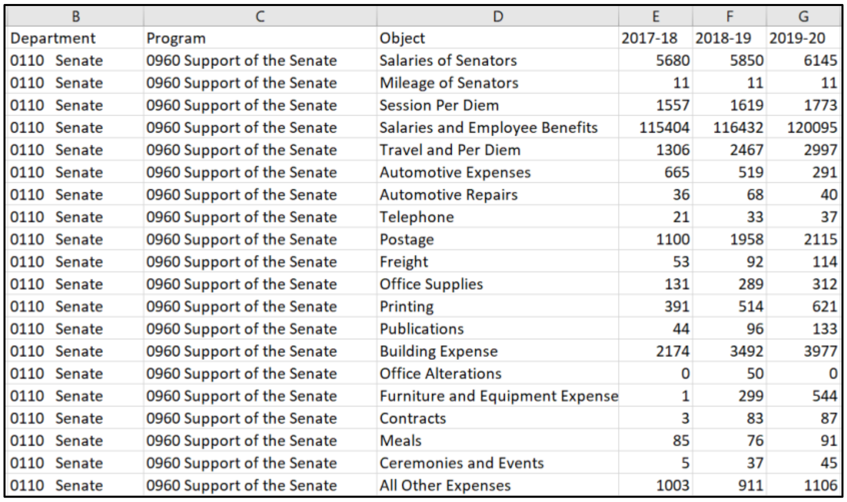
| 3 | | | | Business, | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 100 | | | E | Baseline Positions | | | 13.0 | 13.0 | 13.0 | $1,607 | $1,536 | $1,536 |
| 101 | | | E | Other Adjustments | | | -6.0 | - | - | -698 | 40 | 40 |
| 103 | | | E | Staff Benefits | | | - | - | - | 469 | 805 | 805 |
| 105 | | | E | | OPERATING EXPENSES AND EQUIPMENT | | | | | $540 | $677 | $676 |
| 879 | | | E | | Totals, State Operations | $15,357 | $16,323 | $17,614 | | | | |
| 891 | | | E | | Totals, State Operations | -$26 | -$26 | -$26 | | | | |
| 897 | | | E | | Totals, State Operations | $3,577 | $4,007 | $4,948 | | | | |
| 903 | | | E | | Totals, State Operations | $879 | $1,065 | $1,291 | | | | |
| 915 | | | E | | Totals, State Operations | $1,790 | $1,774 | $1,839 | | | | |
| 920 | | | E | | Totals, State Operations | $55 | $55 | $55 | | | | |
| 931 | | | E | | Totals, State Operations | $84 | $115 | $119 | | | | |
| 937 | | | E | | Totals, State Operations | $11,704 | $12,096 | $12,961 | | | | |
| 943 | | | E | | Totals, State Operations | $4,121 | $5,090 | $4,566 | | | | |
| 950 | | | E | | Totals, State Operations | $20,136 | $20,179 | $20,730 | | | | |
| 965 | | | E | | Totals, State Operations | $66,929 | $70,686 | $78,624 | | | | |
| 970 | | | E | | Totals, State Operations | $92 | $100 | $100 | | | | |
| 977 | | | E | | Totals, State Operations | $3,916 | $2,899 | $3,514 | | | | |

Ready 122 of 7272 records found Count: 122

**Figure 11.** Screenshot of the EXCEL version of report 1000 labeled and ready to be used for scraping. Analysts labeled 122 objects whose expenditure values were to be collected via scraping.

### 4.4 STATEFIN Scraping

The scraping program leverages the Pandas Python module to ingest the labeled EXCEL as a DataFrame and read it row by row. For each row, it checks for a label for either a department, program, or object. First, a department name needs to have been detected and collected before the program can find either a program or an object directly. After a department has been found, the program will only check for programs or objects in the subsequent rows. If an object is found before a program, it will collect the object's name and dollar value and assume an empty program name. Otherwise, if a program is found, it will collect the program name and start looking for objects in the following rows. Figure 12 shows an example of the final output in EXCEL format.

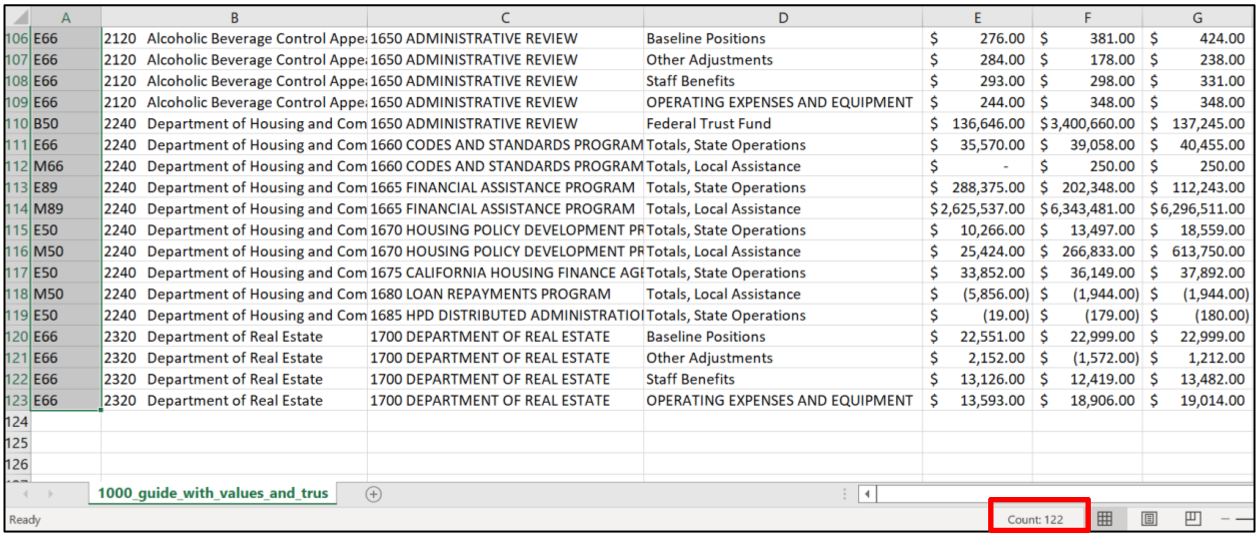


| B | C | D | E | F | G |
|---|---|---|---|---|---|
| Department | Program | Object | 2017-18 | 2018-19 | 2019-20 |
| 0110 Senate | 0960 Support of the Senate | Salaries of Senators | 5680 | 5850 | 6145 |
| 0110 Senate | 0960 Support of the Senate | Mileage of Senators | 11 | 11 | 11 |
| 0110 Senate | 0960 Support of the Senate | Session Per Diem | 1557 | 1619 | 1773 |
| 0110 Senate | 0960 Support of the Senate | Salaries and Employee Benefits | 115404 | 116432 | 120095 |
| 0110 Senate | 0960 Support of the Senate | Travel and Per Diem | 1306 | 2467 | 2997 |
| 0110 Senate | 0960 Support of the Senate | Automotive Expenses | 665 | 519 | 291 |
| 0110 Senate | 0960 Support of the Senate | Automotive Repairs | 36 | 68 | 40 |
| 0110 Senate | 0960 Support of the Senate | Telephone | 21 | 33 | 37 |
| 0110 Senate | 0960 Support of the Senate | Postage | 1100 | 1958 | 2115 |
| 0110 Senate | 0960 Support of the Senate | Freight | 53 | 92 | 114 |
| 0110 Senate | 0960 Support of the Senate | Office Supplies | 131 | 289 | 312 |
| 0110 Senate | 0960 Support of the Senate | Printing | 391 | 514 | 621 |
| 0110 Senate | 0960 Support of the Senate | Publications | 44 | 96 | 133 |
| 0110 Senate | 0960 Support of the Senate | Building Expense | 2174 | 3492 | 3977 |
| 0110 Senate | 0960 Support of the Senate | Office Alterations | 0 | 50 | 0 |
| 0110 Senate | 0960 Support of the Senate | Furniture and Equipment Expense | 1 | 299 | 544 |
| 0110 Senate | 0960 Support of the Senate | Contracts | 3 | 83 | 87 |
| 0110 Senate | 0960 Support of the Senate | Meals | 85 | 76 | 91 |
| 0110 Senate | 0960 Support of the Senate | Ceremonies and Events | 5 | 37 | 45 |
| 0110 Senate | 0960 Support of the Senate | All Other Expenses | 1003 | 911 | 1106 |

**Figure 12.** Screenshot of final scraped output in EXCEL. This EXCEL organizes the state expenditure values by agency, department, program, object, and year.

### 4.5 STATEFIN Results

The final count from the scraped data for report 1000 matches 100% of the labeled objects. This means each of the 122 state finance expenditure objects for this report that was labeled for scraping was found and its value captured correctly. See Figure 13 for a corresponding screenshot.



|  | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| 106 | E66 | 2120 Alcoholic Beverage Control Appe: | 1650 ADMINISTRATIVE REVIEW | Baseline Positions | $ 276.00 | $ 381.00 | $ 424.00 |
| 107 | E66 | 2120 Alcoholic Beverage Control Appe: | 1650 ADMINISTRATIVE REVIEW | Other Adjustments | $ 284.00 | $ 178.00 | $ 238.00 |
| 108 | E66 | 2120 Alcoholic Beverage Control Appe: | 1650 ADMINISTRATIVE REVIEW | Staff Benefits | $ 293.00 | $ 298.00 | $ 331.00 |
| 109 | E66 | 2120 Alcoholic Beverage Control Appe: | 1650 ADMINISTRATIVE REVIEW | OPERATING EXPENSES AND EQUIPMENT | $ 244.00 | $ 348.00 | $ 348.00 |
| 110 | B50 | 2240 Department of Housing and Com | 1650 ADMINISTRATIVE REVIEW | Federal Trust Fund | $ 136,646.00 | $ 3,400,660.00 | $ 137,245.00 |
| 111 | E66 | 2240 Department of Housing and Com | 1660 CODES AND STANDARDS PROGRAM | Totals, State Operations | $ 35,570.00 | $ 39,058.00 | $ 40,455.00 |
| 112 | M66 | 2240 Department of Housing and Com | 1660 CODES AND STANDARDS PROGRAM | Totals, Local Assistance | $ - | $ 250.00 | $ 250.00 |
| 113 | E89 | 2240 Department of Housing and Com | 1665 FINANCIAL ASSISTANCE PROGRAM | Totals, State Operations | $ 288,375.00 | $ 202,348.00 | $ 112,243.00 |
| 114 | M89 | 2240 Department of Housing and Com | 1665 FINANCIAL ASSISTANCE PROGRAM | Totals, Local Assistance | $ 2,625,537.00 | $ 6,343,481.00 | $ 6,296,511.00 |
| 115 | E50 | 2240 Department of Housing and Com | 1670 HOUSING POLICY DEVELOPMENT PR | Totals, State Operations | $ 10,266.00 | $ 13,497.00 | $ 18,559.00 |
| 116 | M50 | 2240 Department of Housing and Com | 1670 HOUSING POLICY DEVELOPMENT PR | Totals, Local Assistance | $ 25,424.00 | $ 266,833.00 | $ 613,750.00 |
| 117 | E50 | 2240 Department of Housing and Com | 1675 CALIFORNIA HOUSING FINANCE AGI | Totals, State Operations | $ 33,852.00 | $ 36,149.00 | $ 37,892.00 |
| 118 | M50 | 2240 Department of Housing and Com | 1680 LOAN REPAYMENTS PROGRAM | Totals, Local Assistance | $ (5,856.00) | $ (1,944.00) | $ (1,944.00) |
| 119 | E50 | 2240 Department of Housing and Com | 1685 HPD DISTRIBUTED ADMINISTRATIOI | Totals, State Operations | $ (19.00) | $ (179.00) | $ (180.00) |
| 120 | E66 | 2320 Department of Real Estate | 1700 DEPARTMENT OF REAL ESTATE | Baseline Positions | $ 22,551.00 | $ 22,999.00 | $ 22,999.00 |
| 121 | E66 | 2320 Department of Real Estate | 1700 DEPARTMENT OF REAL ESTATE | Other Adjustments | $ 2,152.00 | $ (1,572.00) | $ 1,212.00 |
| 122 | E66 | 2320 Department of Real Estate | 1700 DEPARTMENT OF REAL ESTATE | Staff Benefits | $ 13,126.00 | $ 12,419.00 | $ 13,482.00 |
| 123 | E66 | 2320 Department of Real Estate | 1700 DEPARTMENT OF REAL ESTATE | OPERATING EXPENSES AND EQUIPMENT | $ 13,593.00 | $ 18,906.00 | $ 19,014.00 |
| 124 |  |  |  |  |  |  |  |
| 125 |  |  |  |  |  |  |  |
| 126 |  |  |  |  |  |  |  |

**Figure 13.** Screenshot of final scraped data for report 1000. All 122 labeled objects were found and their values captured.

Scraping results for report 0010 totaled 306 out of 306 labeled objects, which means a match rate of 100% as well. These are terrific results for capturing the dollar values precisely for each combination of department, program, and object from reports 1000 and 0010. However, there is still a need to decrease the amount of manual labeling for next iterations and to improve the scraping method so it can be applied successfully to other reports.

## 5. Conclusions and Future Work

This report described two web scraping efforts in support of the Census Bureau's public sector programs. The QTAX project involves scraping a relatively small number of tax revenue items from 26 state government websites. The STATEFIN project, on the other hand, is more focused. It is currently in a prototype stage and involves scraping many financial items from a limited number of documents on the California state government website. Results so far for both QTAX and STATEFIN are positive. The downloading and scraping methodologies are flexible and have shown they can handle inconsistencies in URLs and document layouts. Refinement is needed, but much progress has been made towards automating a substantial portion of what used to be manual work.

The overall goal of these efforts is to get the methodology to the point where it can be put into production and integrated with existing survey processes. Based on the encouraging results obtained so far for QTAX and STATEFIN, there is also potential in applying similar methodology to other public sector surveys. For example, other surveys of state and local governments collect data on public retirement systems and employment and payroll. Lastly, in an effort to be transparent with respondents and the greater public, there are plans to update the publicly available SABLE GitHub repository with the most recent Python code. This repository is located at https://github.com/uscensusbureau/SABLE and received its last major update after the initial PDF methods for QTAX were developed.

## Acknowledgments

The authors would like to thank Kristina Pasquino-Frates, Stephen Owens, Kimberley Moore, Stephen Kaputa, and Amy Newman-Smith of the U.S. Census Bureau for reviewing this paper and providing helpful comments.

## References

The Apache Software Foundation. (2022). Apache Nutch. <http://nutch.apache.org>. Accessed August 15, 2022.

Ariga, A. (2019). tabula-py: Read tables in a PDF into DataFrame. <https://tabula-py.readthedocs.io/en/latest/>. Accessed August 15, 2022.

Bird, S. (2006). NLTK: The Natural Language Toolkit. *Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions*. Sydney, Australia: Association for Computational Linguistics, 69–72.

Crummy. (2022). Beautiful Soup. <https://www.crummy.com/software/BeautifulSoup>. Accessed September 12, 2022.

Dumbacher, B. and Diamond, L.K. (2018). SABLE: Tools for Web Crawling, Web Scraping, and Text Classification. *Proceedings of the 2018 Federal Committee on Statistical Methodology (FCSM) Research Conference*. Washington, DC: Federal Committee on Statistical Methodology.

Dumbacher, B. and Hanna, D. (2017). Using Passive Data Collection, System-to-System Data Collection, and Machine Learning to Improve Economic Surveys. *2017 Proceedings of the American Statistical Association, Business and Economic Statistics Section*. Alexandria, VA: American Statistical Association, 772–785.

McKinney, W. (2010). Data Structures for Statistical Computing in Python. *Proceedings of the 9th Python in Science Conference*, 56–61. <https://www.doi.org/10.25080/Majora-92bf1922-00a>. Accessed September 12, 2022.

Mitchell, R. (2015). *Web Scraping with Python: Collecting Data from the Modern Web*. Sebastopol, CA: O'Reilly Media, Inc.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, *12*, 2825-2830.

Shinyama, Y. (2013). PDFMiner. <http://www.unixuser.org/~euske/python/pdfminer/index.html>. Accessed August 15, 2022.

U.S. Census Bureau. (2022a). Public Sector. <https://www.census.gov/topics/public-sector.html>. Accessed August 2, 2022.

U.S. Census Bureau. (2022b). Quarterly Summary of State & Local Tax Revenue (QTAX). <https://www.census.gov/programs-surveys/qtax.html>. Accessed August 3, 2022.

U.S. Census Bureau. (2022c). Annual Survey of State Government Finances. <https://www.census.gov/programs-surveys/state.html>. Accessed August 3, 2022.

## **Appendix A**: Example Python Code

**Code Block 1**: Example Code for Section 3.1 QTAX PDF Downloading

```
targetPDFNames = []
targetURLs = []
url = "https://dor.ms.gov/statistics/cash-reports"
req = Request(url, headers={'User-Agent': 'SABLE (U.S. Census Bureau
    research to find alternative data sources and reduce respondent
    burden) https://github.com/uscensusbureau/sable/; census-aidcrb-
    support-team@census.gov; For more information, go to
    www.census.gov/scraping/'})
page=urlopen(req).read()
html = page.decode("utf-8")
soup=BeautifulSoup(html,"lxml")
link=""
tables=soup.find_all("div",attrs={"class":" views-field-title"})
for t in tables:
    row=t.find("tr")
    cols=row.find_all("th")
    if "{} - {}".format(mm,month) in cols[0].text:
        link = "https://dor.ms.gov/" + cols[1].find("a").get("href")
        break
targetURLs.append(link)
targetPDFNames.append(link[link.rfind("/")+1:-4])
return targetPDFNames, targetURLs
```

**Code Block 2**: Example Code for Section 3.4 QTAX EXCEL Scraping

```
raw_excel = pd.read_excel(rawExcelLoc, sheetname=tab, header=4)
col_name = month.upper() +" " + yyyy
raw_excel = raw_excel.set_index('Unnamed: 0')
excel_col = raw_excel[col_name].iteritems()
# Finding tax types, values, and tax codes
final_excel = {
    'state':[],
    'year':[],
    'month':[],
    'tax_type':[],
    'tax_value':[],
    'tax_unit':[],
    'tax_time':[],
    'tax_code':[]
}
for index, value in excel_col:
    if type(index) == type(""):
        index_clean = index.rstrip()
        index_clean = index_clean.lstrip()
        index_clean = index_clean.lower()
        for index2, tax_type in tax_codes["tax_type"].iteritems():
            if index_clean == tax_type:
                final_excel["state"].append(state)
                final_excel["year"].append(yyyy)
                final_excel["month"].append(mm)
                final_excel["tax_type"].append(index_clean)
                final_excel["tax_value"].append(value)
                final_excel["tax_unit"].append("dollars")
                final_excel["tax_time"].append("month")
                tax_code =  tax_codes["tax_code"].iloc[index2]
                final_excel["tax_code"].append(tax_code)
final_excel_df = pd.DataFrame.from_dict(final_excel)
```