README

LE: 31 July 2023

[Purpose of this project 2](#_Toc141737737)

[Structure of this project 3](#_Toc141737738)

[historic: 3](#_Toc141737739)

[ weekly\_data: 3](#_Toc141737740)

[ roundup\_scripts 4](#_Toc141737741)

[o scrapers 4](#_Toc141737742)

[I don’t care about this stuff. How do I actually run the project? 5](#_Toc141737743)

[Additional comments and to-dos. 5](#_Toc141737744)

[Comments James 5](#_Toc141737745)

# Purpose of this project

The purpose of this project is to scrape various websites where economics working papers – papers that haven’t been peer reviewed yet – are published, so that the newest research can be tracked and aggregated. The basic project runs several Python scripts that each scrape a specific website and extract data: The title of the paper, its author(s), the date it was posted, a link to the text, an ID number (using the numbering system of the publishing website), and an abstract. Data from all sources are then combined and compared to previous scrapes. Those papers that haven’t been seen before – the newest papers – are outputted to a .csv file which can be easily opened in Excel.

Websites that are scraped for data, as of August 2023, are:

|  |  |
| --- | --- |
| **Name of website** | **Name of script** |
| [Bureau of Economic Analysis](https://www.bea.gov/) | roundup\_scripts/scrapers/BEA.py |
| [Becker Friedman Institute](https://bfi.uchicago.edu/) (at the University of Chicago) | roundup\_scripts/scrapers/BFI.py |
| [Bank for International Settlements](https://www.bis.org/) | roundup\_scripts/scrapers/BIS.py |
| [Federal Reserve Board of Governors](https://www.federalreserve.gov/) (of the United States) | roundup\_scripts/scrapers/Board.py |
| [Bank of England](https://www.bankofengland.co.uk/) | roundup\_scripts/scrapers/BOE.py |
| [Federal Reserve Bank of Chicago](https://www.chicagofed.org/) | roundup\_scripts/scrapers/Chicago.py |
| [European Central Bank](https://www.ecb.europa.eu/home/html/index.en.html) | roundup\_scripts/scrapers/ECB.py |
| [International Monetary Fund](https://www.imf.org/en/Home) | roundup\_scripts/scrapers/IMF.py |
| [National Bureau of Economic Research](https://www.nber.org/) | roundup\_scripts/scrapers/NBER.py |
| [Federal Reserve Bank of New York](https://www.newyorkfed.org/) | roundup\_scripts/scrapers/NewYork.py |

This web scraper was originally developed for use at the [Brookings Institution Hutchins Center on Fiscal and Monetary Policy](https://www.brookings.edu/centers/the-hutchins-center-on-fiscal-and-monetary-policy/the-hutchins-roundup/). The center publishes a weekly newsletter called the [Hutchins Roundup](https://www.brookings.edu/centers/the-hutchins-center-on-fiscal-and-monetary-policy/the-hutchins-roundup/) which summarizes three new economics papers, as well as featuring a chart and recent quote from a famous macroeconomist. (Everyone should subscribe to it – it’s a great email!) The newest papers are examined on a weekly basis, and the scholars at the center make a call on which three should be included in the newsletter.

# Structure of this project

All the project contents are contained in a main folder called “roundup”. The schematic below illustrates the basic file structure of the project. Detailed descriptions are also written below, in a hierarchical list.

## historic:

A folder containing data that has been previously scraped in this project.

## weekly\_data:

A folder containing the data that is gathered in every scrape of the project. Files are stored in the format YYYY-MM-DD-HHMM.csv and YYYY-MM-DD-HHMM.txt for the time the code was run. The .csv files contain the actual data that was newly seen in a given run of runall.py: the title, abstract, author, number, date, etc. This new data can easily be viewed using Microsoft Excel. But even better for viewing are the .html files, which can easily be browsed through. The .txt files are intended more for reference. They contain only the ID numbers of the of the novel data contained in the .csv and .html files.

* papers\_we\_have\_seen.txt:

A file that can be considered the main historical record of the project. It tells compare.py which papers we have seen and which we haven’t by storing all of the index numbers of the papers that have been seen as a python set. Note that no data is stored here aside from index numbers (this is a memory saving feature of the repository).

## roundup\_scripts

A folder containing all of the code used in the project, except for troubleshooter.py and runall.py.

* + compare.py:

A script that contains a function used within runall.py called compare\_historic(df). The function compare\_historical.df takes the most recently scraped data frame (df) as its only input argument and compares it to the data in papers\_we\_have\_seen.txt. It then only “keeps” the papers that are newly seen and saves the new data in historic/weekly\_data/YYYY-MM-DD-HHMM.csv and historic/weekly\_data/YYYY-MM-DD-HHMM.txt files.

## scrapers

A folder that contains each of the individual web scrapers that goes to a specific website like BIS, Chicago Fed, NBER, etc. The scripts are named accordingly. All scripts have analogous functions called scrape(). These functions scrape their respective websites – so, for example, in runall.py, we can import BIS and run BIS.scrape() to get the most recent data (formatted as a pandas data frame) from the Bank for International Settlements, or we can import NBER and run NBER.scrape() to get a data frame of the most recent data scraped from the National Bureau of Economic Research.

* runall.py

The main script in this project. It loops through each of the scripts in roundup\_scripts/scrapers/XXX.py, gathering a data frame of all of the new data available from each website. Then it invokes the compare\_historic(df) function from roundup\_scripts/compare.py to see which of the scripts have already been seen, and which are truly novel. Compare\_historic(df) uses data from papers\_we\_have\_seen.txt to make this determination. Once compare\_historic(df) has been successfully executed, new date- and timestamped files are saved as historic/weekly\_data/YYYY-MM-DD-HHMM.csv and historic/weekly\_data/YYYY-MM-DD-HHMM.txt which contain the truly novel information.

* troubleshooter.py

A script Lorae is currently using on occasion to troubleshoot her code. Should she instead get vscode so she is not using Notepad++ and IDLE? Probably. But for now, this works.

* README.docx

The document you are currently reading.

* Python Tutorial v2.docx

A tutorial for new Python users trying to get set up on this project.

* Requirements.txt

The necessary file to get your venv set up on this project.

# I don’t care about this stuff. How do I actually run the project?

Sorry, these instructions are only for Windows (I’m not sure how Mac works)

1. Download this repository. If you’re reading this, you’ve probably already done it.
2. Open PowerShell. Change your working directory to the working directory of the project, like this:

cd(“C:/path/to/roundup”)

1. Setup the venv for the project using requirements.txt. You will only have to do this once. (If this step confuses you, please reference Python Tutorial v2.docx for more information.)
2. Activate the venv.

venv/Scripts/activate

* OPTIONAL STEP: Now, open runall.py in your favorite text editor. Check that line 26 (or somewhere around there) matches the path to the venv you’ve set up. If you followed my directions on step 1-4 and use a Windows computer, no changes are needed. But if you run into strange errors with modules and things not loading, note that the issue is likely here.

# Path to venv python

venv\_python\_path = " venv/Scripts/python.exe"

1. Go back to PowerShell. With the venv active, run the script runall.py:  
   python runall.py

If this doesn’t work, try typing “python3” instead of “python”.

In summary, it takes only 3 lines of code in PowerShell (after the venv has been set up):

cd(“C:/path/to/roundup”)

venv/Scripts/activate

python runall.py

Results of your run will be located in historic/weekly\_data/YYYY-MM-DD-HHMM.csv.

# Additional comments and to-dos.

## To dos

* Aug 1, 2023: ECB is messed up – the author does not always contain the authors (sometimes it has the title of the paper and other information in it).
* Aug 1, 2023: Standardize the dates to a certain format so that eventually the data can be filtered on them
* DONE 8/15/2023: Aug 3, 2023: Do FRB SF. Note that it is java rendered and there’s no RSS feed for the working papers.  
  <https://www.frbsf.org/economic-research/publications/working-papers/>
  + And boston. Same story: <https://www.bostonfed.org/publications-and-data.aspx#/search?yrFrom=&yrTo=&vol=&series=8c2cb55e251349c59d19db1040c20368&yr=2023>  
    <https://www.bostonfed.org/publications/research-department-working-paper/>
  + Boston had an RSS feed
* DONE 8/11/2023Aug 10, 2023: Do Fed Notes: <https://www.federalreserve.gov/econres/notes/feds-notes/the-restrained-recovery-of-state-and-local-government-payrolls-from-the-pandemic-recession-20230804.html>
* Aug 15, 2023: DONE SF FED 8/21/2023Convert the SF Fed and the Cleveland Fed ones into something more similar to NY Fed – instead of using Selenium, use whatever requests package it is to get the information from the Java rendered webpage.

## Comments James

Great script – very impressive!!!

* On Mac, file paths are slightly different when setting venv path. Might be helpful to detail this in the readme.
* Spyder on my computer must have something else named “scripts”, because when I tried to import compare\_new\_data, I kept running into a ModuleNotFound error. This was fixed when I changed the “scripts” folder name to “roundup\_scripts”. I didn’t get this error when I ran directly from terminal, however.
* There was a merging issue in line 29 under data\_compare.py. If you specifiy right\_on and left\_on to “Number”, this fixes the issue (not sure what it was trying to merge on before).
* There some issues with the code when you tried to merge the hist and recent df’s, so I replaced it with the pd.concat function and filtered out duplicate numbers.