Efficient Literature Searches using Python

Blair Bilodeau

May 30, 2020

University of Toronto & Vector Institute

VECTOR INSTITUTE

CCÉS CSSC

Société statistique du Canada
Statistical Society of Canada

Workshop Motivation

- Me trying to read all the new papers posted on arXiv

Workshop Goals

Workshop Goals

- Discuss the goal of focused literature searches v.s. reading new updates.

Workshop Goals

- Discuss the goal of focused literature searches v.s. reading new updates.
  - At what stage of a project is one more appropriate than another?
  - Which tools are more suited to one over the other?

Workshop Goals

- Discuss the goal of focused literature searches v.s. reading new updates.
  - At what stage of a project is one more appropriate than another?
  - Which tools are more suited to one over the other?
- Learn how to install and get setup using Python.

Workshop Goals

- Discuss the goal of focused literature searches v.s. reading new updates.
  - At what stage of a project is one more appropriate than another?
  - Which tools are more suited to one over the other?
- Learn how to install and get setup using Python.
  - This will be quick, just to get everyone on the same page.

Workshop Goals

- Discuss the goal of focused literature searches v.s. reading new updates.
  - At what stage of a project is one more appropriate than another?
  - Which tools are more suited to one over the other?
- Learn how to install and get setup using Python.
  - This will be quick, just to get everyone on the same page.
- Learn how to write a Python script to scrape arXiv and biorXiv papers.

Workshop Goals

- Discuss the goal of focused literature searches v.s. reading new updates.
  - At what stage of a project is one more appropriate than another?
  - Which tools are more suited to one over the other?
- Learn how to install and get setup using Python.
  - This will be quick, just to get everyone on the same page.
- Learn how to write a Python script to scrape arXiv and biorXiv papers.
  - Cover the basics (libraries, functions, some syntax).
  - Explore customization options for the script.

Workshop Goals

- Discuss the goal of focused literature searches v.s. reading new updates.
  - At what stage of a project is one more appropriate than another?
  - Which tools are more suited to one over the other?
- Learn how to install and get setup using Python.
  - This will be quick, just to get everyone on the same page.
- Learn how to write a Python script to scrape arXiv and biorXiv papers.
  - Cover the basics (libraries, functions, some syntax).
  - Explore customization options for the script.
- Automate the running of this script.

Workshop Goals

- Discuss the goal of focused literature searches v.s. reading new updates.
  - At what stage of a project is one more appropriate than another?
  - Which tools are more suited to one over the other?
- Learn how to install and get setup using Python.
  - This will be quick, just to get everyone on the same page.
- Learn how to write a Python script to scrape arXiv and biorXiv papers.
  - Cover the basics (libraries, functions, some syntax).
  - Explore customization options for the script.
- Automate the running of this script.
  - Running from the command line.
  - Scheduling the script to run at certain times.

Workshop Goals

- Discuss the goal of focused literature searches v.s. reading new updates.
  - At what stage of a project is one more appropriate than another?
  - Which tools are more suited to one over the other?
- Learn how to install and get setup using Python.
  - This will be quick, just to get everyone on the same page.
- Learn how to write a Python script to scrape arXiv and biorXiv papers.
  - Cover the basics (libraries, functions, some syntax).
  - Explore customization options for the script.
- Automate the running of this script.
  - Running from the command line.
  - Scheduling the script to run at certain times.
- Practice!

Large Literature Searches v.s. Daily Updates

Large Literature Searches v.s. Daily Updates

Large Literature Searches

Large Literature Searches v.s. Daily Updates

Large Literature Searches

- Understand the history of a topic.

Large Literature Searches v.s. Daily Updates

Large Literature Searches

- Understand the history of a topic.
- Identify which problems have been solved and which remain open.

Large Literature Searches v.s. Daily Updates

Large Literature Searches

- Understand the history of a topic.
- Identify which problems have been solved and which remain open.
- Curate a large collection of fundamental literature which can be drawn from for multiple projects.

Large Literature Searches v.s. Daily Updates

Large Literature Searches

- Understand the history of a topic.
- Identify which problems have been solved and which remain open.
- Curate a large collection of fundamental literature which can be drawn from for multiple projects.

Daily Updates

Large Literature Searches v.s. Daily Updates

Large Literature Searches

- Understand the history of a topic.
- Identify which problems have been solved and which remain open.
- Curate a large collection of fundamental literature which can be drawn from for multiple projects.

Daily Updates

- Find papers which might help you solve your current problem.

Large Literature Searches v.s. Daily Updates

Large Literature Searches

- Understand the history of a topic.
- Identify which problems have been solved and which remain open.
- Curate a large collection of fundamental literature which can be drawn from for multiple projects.

Daily Updates

- Find papers which might help you solve your current problem.
- Find papers which inspire future projects to start thinking about.

Large Literature Searches v.s. Daily Updates

Large Literature Searches

- Understand the history of a topic.
- Identify which problems have been solved and which remain open.
- Curate a large collection of fundamental literature which can be drawn from for multiple projects.

Daily Updates

- Find papers which might help you solve your current problem.
- Find papers which inspire future projects to start thinking about.
- Find out if you've been scooped.

Large Literature Searches v.s. Daily Updates

Large Literature Searches

- Understand the history of a topic.
- Identify which problems have been solved and which remain open.
- Curate a large collection of fundamental literature which can be drawn from for multiple projects.

Daily Updates

- Find papers which might help you solve your current problem.
- Find papers which inspire future projects to start thinking about.
- Find out if you've been scooped.
- Avoid keeping track of all new papers – many will quickly become irrelevant to you.

Preprint Servers

Preprint Servers

Researchers in stats, math, bio, medicine, and other fields use these to post versions of papers before publication (as well as open source access after publication).

`https://arxiv.org`
`https://www.biorxiv.org`
`https://www.medrxiv.org`

Preprint Servers

Researchers in stats, math, bio, medicine, and other fields use these to post versions of papers before publication (as well as open source access after publication).

`https://arxiv.org`
`https://www.biorxiv.org`
`https://www.medrxiv.org`

Advantages

Preprint Servers

Researchers in stats, math, bio, medicine, and other fields use these to post versions of papers before publication (as well as open source access after publication).

`https://arxiv.org`
`https://www.biorxiv.org`
`https://www.medrxiv.org`

Advantages

- Expands visibility/accessibility of papers.

Preprint Servers

Researchers in stats, math, bio, medicine, and other fields use these to post versions of papers before publication (as well as open source access after publication).

`https://arxiv.org`
`https://www.biorxiv.org`
`https://www.medrxiv.org`

Advantages

- Expands visibility/accessibility of papers.
- Allows for feedback from the community in addition to journal reviewers.

Preprint Servers

Researchers in stats, math, bio, medicine, and other fields use these to post versions of papers before publication (as well as open source access after publication).

`https://arxiv.org`
`https://www.biorxiv.org`
`https://www.medrxiv.org`

Advantages

- Expands visibility/accessibility of papers.
- Allows for feedback from the community in addition to journal reviewers.
- Mitigates chances of getting scooped during long journal revision times.

Preprint Servers

Researchers in stats, math, bio, medicine, and other fields use these to post versions of papers before publication (as well as open source access after publication).

`https://arxiv.org`
`https://www.biorxiv.org`
`https://www.medrxiv.org`

Advantages

- Expands visibility/accessibility of papers.
- Allows for feedback from the community in addition to journal reviewers.
- Mitigates chances of getting scooped during long journal revision times.

Disadvantages

Preprint Servers

Researchers in stats, math, bio, medicine, and other fields use these to post versions of papers before publication (as well as open source access after publication).

`https://arxiv.org`
`https://www.biorxiv.org`
`https://www.medrxiv.org`

Advantages

- Expands visibility/accessibility of papers.
- Allows for feedback from the community in addition to journal reviewers.
- Mitigates chances of getting scooped during long journal revision times.

Disadvantages

- No peer-review, so papers may be rougher.

Preprint Servers

Researchers in stats, math, bio, medicine, and other fields use these to post versions of papers before publication (as well as open source access after publication).

`https://arxiv.org`
`https://www.biorxiv.org`
`https://www.medrxiv.org`

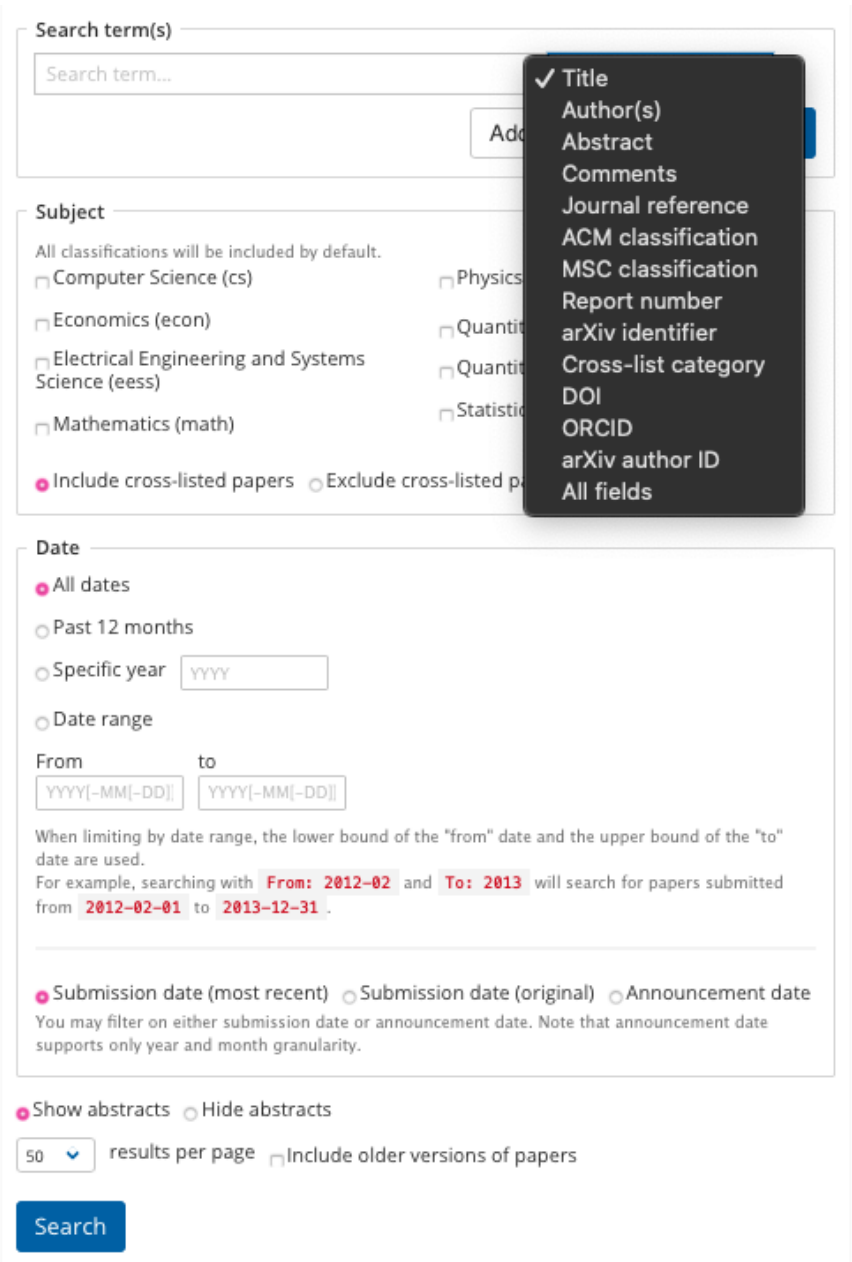
Advantages

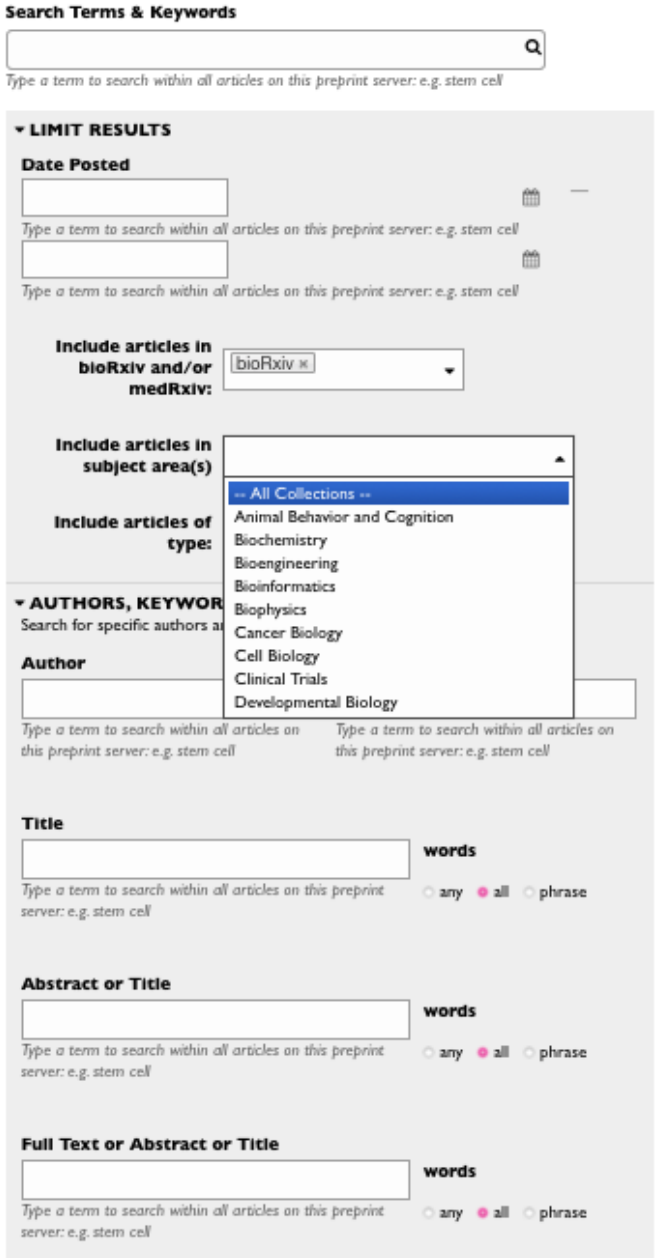
- Expands visibility/accessibility of papers.
- Allows for feedback from the community in addition to journal reviewers.
- Mitigates chances of getting scooped during long journal revision times.

Disadvantages

- No peer-review, so papers may be rougher.
- Easy to get lost in a sea of papers.

Preprint Server Search Options





Existing Automation Options

Existing Automation Options

Arxiv Email Alerts (`https://arxiv.org/help/subscribe`)

Existing Automation Options

Arxiv Email Alerts (`https://arxiv.org/help/subscribe`)

- Daily email with titles and abstracts of all paper uploads in a specific subject.

Existing Automation Options

Arxiv Email Alerts (`https://arxiv.org/help/subscribe`)

- Daily email with titles and abstracts of all paper uploads in a specific subject.
- No ability to filter by search terms.

Existing Automation Options

Arxiv Email Alerts (`https://arxiv.org/help/subscribe`)

- Daily email with titles and abstracts of all paper uploads in a specific subject.
- No ability to filter by search terms.

Arxiv Sanity Preserver (`http://www.arxiv-sanity.com`)

Existing Automation Options

Arxiv Email Alerts (`https://arxiv.org/help/subscribe`)

- Daily email with titles and abstracts of all paper uploads in a specific subject.
- No ability to filter by search terms.

Arxiv Sanity Preserver (`http://www.arxiv-sanity.com`)

- Nicer user interface for papers.

Existing Automation Options

Arxiv Email Alerts (`https://arxiv.org/help/subscribe`)

- Daily email with titles and abstracts of all paper uploads in a specific subject.
- No ability to filter by search terms.

Arxiv Sanity Preserver (`http://www.arxiv-sanity.com`)

- Nicer user interface for papers.
- Some text processing to recommend papers.

Existing Automation Options

Arxiv Email Alerts (`https://arxiv.org/help/subscribe`)

- Daily email with titles and abstracts of all paper uploads in a specific subject.
- No ability to filter by search terms.

Arxiv Sanity Preserver (`http://www.arxiv-sanity.com`)

- Nicer user interface for papers.
- Some text processing to recommend papers.
- No automation capabilities.
  (see `https://github.com/MichalMalyska/Arxiv_Sanity_Downloader`)

Existing Automation Options

Arxiv Email Alerts (`https://arxiv.org/help/subscribe`)

- Daily email with titles and abstracts of all paper uploads in a specific subject.
- No ability to filter by search terms.

Arxiv Sanity Preserver (`http://www.arxiv-sanity.com`)

- Nicer user interface for papers.
- Some text processing to recommend papers.
- No automation capabilities.
  (see `https://github.com/MichalMalyska/Arxiv_Sanity_Downloader`)
- Only applies to a few subject fields (machine learning).

Existing Automation Options

Arxiv Email Alerts (`https://arxiv.org/help/subscribe`)

- Daily email with titles and abstracts of all paper uploads in a specific subject.
- No ability to filter by search terms.

Arxiv Sanity Preserver (`http://www.arxiv-sanity.com`)

- Nicer user interface for papers.
- Some text processing to recommend papers.
- No automation capabilities.
  (see `https://github.com/MichalMalyska/Arxiv_Sanity_Downloader`)
- Only applies to a few subject fields (machine learning).

Biorxiv Options

Existing Automation Options

Arxiv Email Alerts (`https://arxiv.org/help/subscribe`)

- Daily email with titles and abstracts of all paper uploads in a specific subject.
- No ability to filter by search terms.

Arxiv Sanity Preserver (`http://www.arxiv-sanity.com`)

- Nicer user interface for papers.
- Some text processing to recommend papers.
- No automation capabilities.
  (see `https://github.com/MichalMalyska/Arxiv_Sanity_Downloader`)
- Only applies to a few subject fields (machine learning).

Biorxiv Options

- No known options to me, besides this project with a broken link.
  (`https://github.com/gokceneraslan/biorxiv-sanity-preserver`)

Customized Python Script

Customized Python Script

Goals

- High flexibility for keyword searching.
- Easy to run and parse output everyday.
- Modular to allow for additional features to be added.

Customized Python Script

Goals

- High flexibility for keyword searching.
- Easy to run and parse output everyday.
- Modular to allow for additional features to be added.

Why Python?

- Easy and fast web-scraping.
- Readable even to a non-programmer.
- I'm familiar with it.

Customized Python Script

Goals

- High flexibility for keyword searching.
- Easy to run and parse output everyday.
- Modular to allow for additional features to be added.

Why Python?

- Easy and fast web-scraping.
- Readable even to a non-programmer.
- I'm familiar with it.

Access the Scripts

`https://github.com/blairbilodeau/arxiv-biorxiv-search`

Downloading Python

Downloading Python

Check if you have it...

Downloading Python

Check if you have it...

- **Mac:** Open "terminal" application and type `python3`
- **Windows:** Open "command prompt" application and type `python` or `python3`

Downloading Python

Check if you have it...

- **Mac:** Open "terminal" application and type `python3`
- **Windows:** Open "command prompt" application and type `python` or `python3`

If you don't see the following, you have to install.

```
blairbilodeau@Blairs-MacBook-Pro ~ % python3
Python 3.7.5 (v3.7.5:5c02a39a0b, Oct 14 2019, 18:49:57)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Downloading Python

Check if you have it...

- **Mac:** Open "terminal" application and type `python3`
- **Windows:** Open "command prompt" application and type `python` or `python3`

If you don't see the following, you have to install.

```
blairbilodeau@Blairs-MacBook-Pro ~ % python3
Python 3.7.5 (v3.7.5:5c02a39a0b, Oct 14 2019, 18:49:57)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

If you do see that, great! You're now in a python environment. Either spend some time in there (try typing `print('hello world!')`) or type `exit()` to leave. Take a break for the next slide.

Downloading Python

Option 1: Directly Download Python

Go to `https://www.python.org/downloads/` and download Python 3.
(The actual version doesn't matter as long as it's Python 3.x.x)

Option 2: Use Anaconda

Download from `https://www.anaconda.com/products/individual`.
(Preferable if you aren't familiar with working on the command line)

Downloading Python

Option 1: Directly Download Python

Go to `https://www.python.org/downloads/` and download Python 3.
(The actual version doesn't matter as long as it's Python 3.x.x)

Option 2: Use Anaconda

Download from `https://www.anaconda.com/products/individual`.
(Preferable if you aren't familiar with working on the command line)

Common Troubleshooting Tips

- Make sure the default python path is python3 if you have both installed
- On Windows, add python to your path environment
    - Computer: Properties: Advanced System Settings: Environment Variables: Path: add ";C:\Python36" (or whichever version) to the end

Python Packages

In order to do anything interesting in Python, you need to install "packages". These are scripts people have written so you don't have to reinvent the wheel.

Python Packages

In order to do anything interesting in Python, you need to install "packages". These are scripts people have written so you don't have to reinvent the wheel.

Installing Packages

We will use `pip`, which is automatically included with installations.

To install a package named `name`:
open up terminal or command prompt and type `pip install name`.
On windows, you may need to type something like
`C:\Python36\Scripts\pip install name` or
`C:\Python36\Scripts\pip.exe install name`

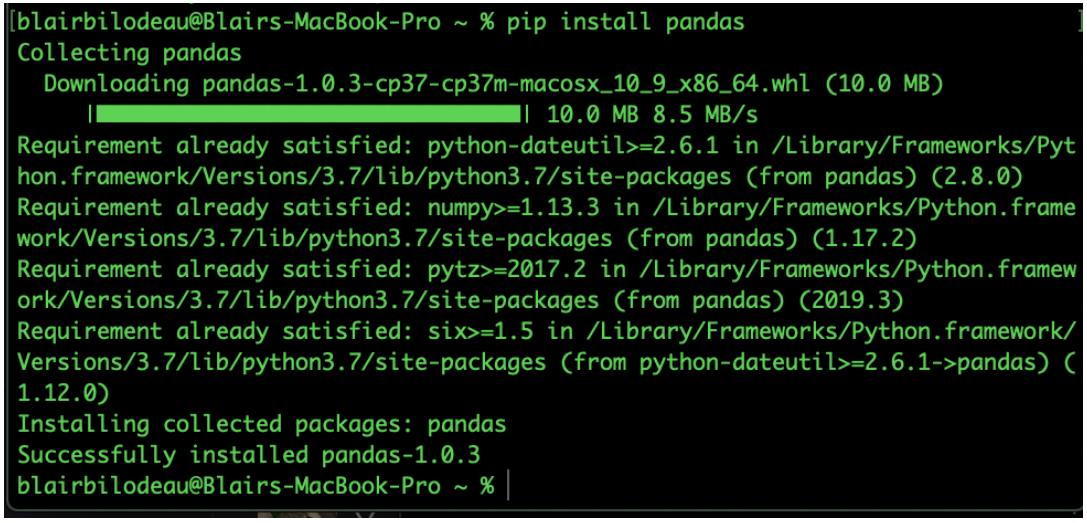
Python Packages

For example, to install the package `pandas`,

```
[blairbilodeau@Blairs-MacBook-Pro ~ % pip install pandas
Collecting pandas
  Downloading pandas-1.0.3-cp37-cp37m-macosx_10_9_x86_64.whl (10.0 MB)
     |████████████████████████████████| 10.0 MB 8.5 MB/s
Requirement already satisfied: python-dateutil>=2.6.1 in /Library/Frameworks/Pyt
hon.framework/Versions/3.7/lib/python3.7/site-packages (from pandas) (2.8.0)
Requirement already satisfied: numpy>=1.13.3 in /Library/Frameworks/Python.frame
work/Versions/3.7/lib/python3.7/site-packages (from pandas) (1.17.2)
Requirement already satisfied: pytz>=2017.2 in /Library/Frameworks/Python.framew
ork/Versions/3.7/lib/python3.7/site-packages (from pandas) (2019.3)
Requirement already satisfied: six>=1.5 in /Library/Frameworks/Python.framework/
Versions/3.7/lib/python3.7/site-packages (from python-dateutil>=2.6.1->pandas) (
1.12.0)
Installing collected packages: pandas
Successfully installed pandas-1.0.3
blairbilodeau@Blairs-MacBook-Pro ~ %
```

Python Packages

For example, to install the package `pandas`,



Extra packages needed for this script...

- `pandas` (data structure tools)
- `requests` (handling opening websites)
- `beautifulsoup4` (parsing HTML)

Scraping from arXiv

Scraping from arXiv

OpenArchive (`https://arxiv.org/help/oa`)

Scraping from arXiv

OpenArchive (`https://arxiv.org/help/oa`)

- Open source initiative to store and provide a coding interface for arXiv.

Scraping from arXiv

OpenArchive (`https://arxiv.org/help/oa`)

- Open source initiative to store and provide a coding interface for arXiv.
- This is used to avoid people remotely making hits on the actual arXiv site.

Scraping from arXiv

OpenArchive (`https://arxiv.org/help/oa`)

- Open source initiative to store and provide a coding interface for arXiv.
- This is used to avoid people remotely making hits on the actual arXiv site.

Script Idea

Scraping from arXiv

OpenArchive (`https://arxiv.org/help/oa`)

- Open source initiative to store and provide a coding interface for arXiv.
- This is used to avoid people remotely making hits on the actual arXiv site.

Script Idea

- Pull all abstracts and titles from OpenArchive within a date range for a specific subject;

Scraping from arXiv

OpenArchive (`https://arxiv.org/help/oa`)

- Open source initiative to store and provide a coding interface for arXiv.
- This is used to avoid people remotely making hits on the actual arXiv site.

Script Idea

- Pull all abstracts and titles from OpenArchive within a date range for a specific subject;
- Check each of these abstract/title combinations against a custom set of keyword matching requirements;

Scraping from arXiv

OpenArchive (`https://arxiv.org/help/oa`)

- Open source initiative to store and provide a coding interface for arXiv.
- This is used to avoid people remotely making hits on the actual arXiv site.

Script Idea

- Pull all abstracts and titles from OpenArchive within a date range for a specific subject;
- Check each of these abstract/title combinations against a custom set of keyword matching requirements;
- Repeat this for each subject;

Scraping from arXiv

OpenArchive (`https://arxiv.org/help/oa`)

- Open source initiative to store and provide a coding interface for arXiv.
- This is used to avoid people remotely making hits on the actual arXiv site.

Script Idea

- Pull all abstracts and titles from OpenArchive within a date range for a specific subject;
- Check each of these abstract/title combinations against a custom set of keyword matching requirements;
- Repeat this for each subject;
- Display the titles and abstracts selected (with other info if desired), with optional exporting of information to `csv` file and downloading of full `pdfs`.

`arxiv_search_function.py` Parameters

```
#######
## Main function
## Parameters:
# sdate        - datetime object        - starting date of search period
# fdate        - datetime object        - end date of search period
# kwd_req      - list of str            - keywords that are required to be in the title or abstract
# kwd_exc      - list of str            - keywords that must not be in the title or abstract
# kwd_one      - list of lists of str   - keywords of which at least one must be included
# athr_req     - list of str            - authors which are required
# athr_exc     - list of str            - authors to exclude
# athr_one     - list of lists of str   - authors of which one must be included
# subjects     - list of str            - subject options are:
#                                           astro-ph, cond-mat, gr-qc, hep-ex, hep-lat, hep-ph, hep-th, math-ph, nlin, nucl-ex, nucl-th,
#                                           physics, quant-ph, math, cs, q-bio, q-fin, stat
# max_records  - int                    - maximum number of results to return
# max_time     - float                  - maximum amount of seconds to be spent searching
# cols         - list                   - arxiv fields to extract
#                                           column options are:
#                                           id, url, title, authors, date, abstract, categories
# export       - str                    - folder location to dump results list in csv
# exportfile   - str                    - by default the file name will be today's date, but you can override this with exportfile
# download     - str                    - folder location to dump returned pdfs into
#                                           (files named using year_lastname format)
```

- `kwd_req, kwd_exc, kwd_one` are the main parameters that allow for custom searching of papers
- All of these are optional – if you don't pass any arguments you will get the first 50 papers from `cs` for the month

`arxiv_search_function.py` **Demonstration**

I will now show an example of running through the code in a Jupyter notebook.

`biorxiv_search_function.py` Parameters

`biorxiv_search_function.py` **Parameters**

- No OpenArchive style API to access papers.

`biorxiv_search_function.py` **Parameters**

- No OpenArchive style API to access papers.
- Instead access their advanced search, which is more limited than the custom search in the arxiv script, but still useful.

`biorxiv_search_function.py` **Parameters**

- No OpenArchive style API to access papers.
- Instead access their advanced search, which is more limited than the custom search in the arxiv script, but still useful.
- The code is then mainly building the URL based on search parameters.

`biorxiv_search_function.py` Parameters

- No OpenArchive style API to access papers.
- Instead access their advanced search, which is more limited than the custom search in the arxiv script, but still useful.
- The code is then mainly building the URL based on search parameters.

```
#######
## Main function
## Parameters:
# sdate        - datetime object        - starting date of search period
# fdate        - datetime object        - end date of search period
# kwd          - list of str            - keywords to search for in title and abstract
# kwd_type     - 'all' or 'any'         - whether all keywords are required or just one of them
# athr         - list of max 2 str      - authors which are required
# subjects     - list of str            - subject options are:
#                                         (Note capitalization and spacing are important for subject)
#                                           Animal Behaviour and Cognition, Biochemistry, Bioengineering, Bioinformatics,
#                                           Biophysics, Cancer Biology, Cell Biology, Clinical Trials, Developmental Biology,
#                                           Ecology, Epidemiology, Evolutionary Biology, Genetics, Genomics, Immunology,
#                                           Microbiology, Molecular Biology, Neuroscience, Paleontology, Pathology,
#                                           Pharmacology and Toxicology, Physiology, Plant Biology, Scientific Communication and Education,
#                                           Synthetic Biology, Systems Biology, Zoology
# max_records  - int                    - maximum number of results to return
# max_time     - float                  - maximum amount of seconds to be spent searching
# cols         - list                   - biorxiv fields to extract
#                                           column options are:
#                                           title, authors, date, url
# abstract     - bool                   - whether to extract the abstract of every paper returned by search
#                                           (potentially very time consuming)
# export       - str                    - folder location to dump results list in csv
# exportfile   - str                    - by default the file name will be today's date, but you can override this with exportfile
# download     - str                    - folder location to dump returned pdfs into
#                                           (files named using year_lastname format)
```

Timesaving Workflow

Running the Script

- Create a separate python file to call the functions with parameters you desire, and run that from command line every day.
- See the file `search_examples.py` in my Github.

Timesaving Workflow

Running the Script

- Create a separate python file to call the functions with parameters you desire, and run that from command line every day.
- See the file `search_examples.py` in my Github.

Automating the Script

- Mac: used `launchd`
  - Create a shell script to run the python file you want (`search_examples.sh`)
  - Place the file `file.name.plist` in /Library/LaunchDaemons with names changed (currently runs once every 24 hours, can be changed).
  - In command line, type `cd Library/LaunchDaemons` and then `sudo launchctl load file.name.plist`.
- Windows: use Task Scheduler
  - Create a batch file to run the python file you want.
  - Follow the instructions in Task Scheduler after clicking `Create Basic Task`.