NCCS Core Files Documentation

Release 1.0

Jeff Levy

Oct 18, 2019

Setup

CHAPTER 1

Introduction

We began rewriting the NCCS core file process in the fall of 2016. The previous system, written in SQL, required manual intervention throughout the build and partial rewriting of the code every year, as well as being a fairly opaque process. It was also part of the NCCS system that was hidden from users, with even the data behind a paywall. We had several goals with this project:

- Create a process that would be consistent from year to year
  - Automate every possible step
  - Streamline any necessary human intervention
- Create a process that was as open as possible
  - Use an open-source code base
  - Make the code publically available
  - Make the final product freely available
- Make our manual changes transparent
- Review some aspects of the data creation that hadn't been looked at in many years
- Reduce the need for maintenance of the code for every new release, thus freeing more NCCS resources to be used on manual validation
- Add some new validation processes, particularly for the form PF

More...

CHAPTER 2

Where to Find the Data

If you only wish to use the final NCCS core file release data, and are not interested in how they were built, you can skip all of this and go directly to the NCCS Data Archive.

CHAPTER 3

Can I Build the Data Myself?

All of our code is open source, and everything you need to build a semblance of the data yourself is available. The only pieces missing are the manual changes NCCS adds each year, which can be significant. See the section on validation below for more details.

If you do not have a login to the NCCS MySQL server (which only works from the Urban campus), you will need to manually download the necessary files first. They are:

- lu_fipsmsa
- nteedocAllEins
- Past core file releases equal to the value set in **main.py** under the **backfill** setting (see below), which is usually the prior three years.

Those files must be placed in the folder **nccs-file-processing\core files\downloads\nccs** as CSV documents.

CHAPTER 4

Building the Core Files

In order to have access to the program state for debugging and the ability to query various intermediate steps after program completion, it is suggested you build the core files from within an interactive interpreter. There are many ways to work with Python that generally come down to personal preference, so you can obviously deviate from these specifics.

Also please note that while the program *should* be entirely platform-agnostic, it has only been tested in a Windows environment.

1. Download the Python version 3.x distribution from Anaconda. The core file process does not support Python version 2.x
2. Install Anaconda, as well as your prefered text editor (or use an option included with Anaconda) if you will want to look at the source code.

- For example, Notepad++ or Atom

3. Clone the NCCS Core File repository from GitHub

- If you do not have GitHub set up, you will need to create a free account and then start by downloading GitHub Desktop
- From the link to the NCCS GitHub repository, there is a green button to the right that says **Clone or download**. Click it, then click **Open in Desktop**.

4. Use your text editor to open **main.py** and set the options at the top, most notably:

- **current_yr**: The release year you want to build.
- **forms**: ['EZ', 'Full', 'PF'] will build all releases.
- **get_from_sql**: Set to False if you do not have a login to the NCCS MySQL server.
  - See the section above, *Can I Build the Data Myself?*

5. Open a command prompt and type in **ipython**.

- On Windows, open a command prompt by pressing **WindowKey+R** and typing **cmd**.
- If the system cannot find ipython and you have just installed Anaconda, you may need to reboot so it updates your system path.

6. Use the **%run** IPython magic to start the build process as follows, substituting in the exact path you cloned the repository to:

```
%run "d:\users\jlevy\documents\github\nccs-file-processing\core␣
↪files\main.py"
```

CHAPTER 5

The Validation Process

The raw form 990 data the IRS releases generally relies upon the submitting firm for the accuracy of the figures. There are three ways the NCCS core file process approaches this issue: checking for arithmetic failures, looking at firms with huge swings in revenue, assets or expenses between years, and paying extra scrutiny to the largest firms who have the most potential to skew aggregate figures.

5.1 Arithmetic Failures

Many parts of the IRS 990 forms ask for the values in a handful of sub fields, followed by a total. Clearly the sub fields should add up to that total, but often times they do not. The core file process tests for this across a handful of columns in all three forms, and marks any that are off by more than a certain threshold (generally $1,000) as "failures".

5.2 Large Changes

Any firm whose revenue, assets or expenses change by more than 50% of their prior-year value will be flagged as a "changed" firm. An additional requirement of revenue over $10 million is imposed, since it is much easier for smaller non profits to experience large percentage swings in these categories.

5.3 Largest Firms

And finally, firms that are in the largest 1% of any of revenue, assets or expenses are marked for review as "large". This is done solely because of their ability to dramatically affect aggregates if anything is incorrect.

The code automatically flags all firms that fall into any of these three categories. However, in most years, NCCS lacks the capacity to manually review every issue. In 2014, the CO, PC and PF release files were flagged with 1,052, 1,314 and 422 issues, respectively. Each of these firms had one or more failures that required the raw IRS data to be manually compared to the firm's actual form 990 (hosted by the Foundation Center), and sometimes other sources as well (such

as the firm's website). This amounted to, in the 2014 CO file, total revenue changes of $1.34 billion, assets of $148 million, and expenses of $40 million.

Many of these organizations flagged as "changed" or "largest" do not end up requiring any adjustments. Sometimes, however, very large errors can be found this way. For example, if you search Foundation Center for EIN **042103594**, you will see the Massachusetts Institute of Technology. This EIN in 2013 has two filings, however; one showing assets of almost $17.7 billion, and the other showing assets of $30,000. A closer examination reveals that the Tau Beta Pi engineering honors society at MIT incorrectly filed their returns using MIT's EIN. This would be clearly flagged as a "changed" firm, and then manually corrected in the NCCS release.

We prioritize the review process by firm size and type of failure, with all of our fixes available for review at (link). Lower-priority failures we are unable to manually review are still marked in the data. The column named **VALIDATION_REASON** can either be empty for firms with no failures, or will be marked **F** for arithmetic failures, **C** for firms with large changes, **L** for the very largest firms, or any permutation of those three together. Compare that to the column **VALIDATION_STATE**, which takes on the value **0** for unreviewed, **1** for fixed, **2** for ignored or **3** for checked okay.

CHAPTER 6

Code Details and Docstrings

Below are the different modules of the core file creation code, along with the descriptions of all of the classes, functions and methods used in each.

6.1 main module

6.2 build_nccs module

6.3 data module

6.4 load_data module

6.5 process module

6.6 process_full module

6.7 process_ez module

6.8 process_pf module

6.9 process_co_pc module

6.10 validate module

6.11 validate_full module

6.12 validate_ez module

6.13 validate_pf module

6.14 write module

6.15 validation_fixer module

CHAPTER 7

Indices and tables

- genindex
- modindex
- search