Automated Data Updates with Python

Nathan Young
NC Data Dashboard
02/01/2020

Contents

Why Automate

As computers continue to get smarter, we felt it was time to look to the future. The previous way of completing updates for the NC Data Dashboard included going to websites, finding data, downloading data, opening Excel to clean the data, export data at a text file to upload to SSMS database. So far, this process used your local machine. Once the text files were created, they had to be moved from your local machine to a server and then could be uploaded to the database. This involved storing data in the cloud via OneDrive or Dropbox. Once multifactored into the server and logged into cloud storage, data could be downloaded onto the server, SSMS could be opened and data could be loaded into database.

This process was very tedious and took a while to accomplish. For example, when working with the Land data, the process from start to the creation of a text file for every notebook took almost 30 minutes. Uploading the data to the dashboard, another 30 minutes.

One hour. To upload one folder of new data. The Dashboard has 6 folders at the time of writing. Now, not every folder contains the same data and some of the processes to get new data are shorter or longer respectively, but it would take hours to update the database. Although we get paid hourly, this had to change.

I started learning Python in college with other members of the Dashboard, but I was seemingly the only one who ran with it. In my free time, I took courses on Pluralsight, DataCamp, Cognitive Class, etc. to learn more and 'beef' up my Python skills. When I had the chance to utilize the skills I had learned, I decided to test my knowledge. I spent hours writing these scripts to make my not only my job easier, but also to propel us, the Dashboard, to new heights.

With updates just a click away, now there is more time to focus on other issues that arise, allow us to find new data, and improve other aspects of the Dashboard.

Going back to the Land folder, updates now take less than 3 minutes and all I must do is click on a file. Productivity is the name of the game. (Total time: less than 8 minutes)

There are still many updates that can and will happen, but it is exciting to be apart of the future success of the NC Data Dashboard.

-Nathan Young

Understanding the Notebooks

All notebooks, no matter the folder, should look and function like each other. This was done to (hopefully) make them easier to understand and use.

Functions of each cell are written as comments in their respective cell, but an overview is here to help comprehension.

- Imports
    - Pandas is a tool used by Python for data analysis
    - ZipFile, os, StringIO, BytesIO, Requests are all used to open zip files
    - sqlalchemy is used to connect Python to database for SQL commands
- Backups
    - This cell takes the previously updated file for the current notebook from the Updates folder, adds _BACKUP to the end the files name, and moves the file to the Backups folder. The only backups will be from the previous update, as they replace each other every update.
- Get new data
    - Each notebook is set to pull data for specific data requested for the Data Dashboard. The pull request is using the web address for the Excel or CSV table found in the source (Land data = Zillow & GeoFRED, Earnings data = BEA, etc.)
- Clean new data
    - In order to be uploaded to the NC Data Dashboard, the data needs to only show North Carolina. To do this, we apply a filter on the State column to show only data where State == NC.
    - MunicipalCodeFIPS is a number but should have 3 digits for all values. Excel removes the leading 0 because nobody reads 031 as 'zero three one'. Its 'thirty-one'.
        - Because there are two- and three-digit values, we need to get the values to match the largest value, three digits, meaning we have to add back the leading 0's.
            - To do this, we convert MunicipalCodeFIPS data type to text with 'str' and then use '.zfill(3)' to fill in the values to 3 places.
- Save new data for export to SSMS
    - To prepare the data for export to the database, it needs to be a tab-delimited text file. 'sep = '\t'' saves the file as tab delimited. Data is now ready to be loaded to SSMS
    - NOTE: SSMS is now updated via dataframe addressed in later steps.

Understanding the Notebooks, Cont.

- Prepare dataframe for upload to SSMS
    - Reset index
        - When we created text files for upload, we had to change the index from row numbers to data for SSMS to read them the way we wanted.
        - SSMS creates its own index which would delete the index we created so we must reset the index.
    - Fill NaN values with 0
        - Missing data in the dataframe is considered ‘NaN’. Tableau team cannot work with ‘NaN’ data so we must convert missing values to ‘0’
- Connect to database
    - Establish connection
        - Connection is made with Windows Authentication so no need to login. ‘Trusted_Connection=yes’ handles login.
        - Server should be whatever server you are on.
        - Database should typically be STG2
    - ‘Autocommit = True’ will commit all SQL code executed by cursor
    - cursor allows execution of SQL scripts in Python
- Clean database
    - Remove old backup tables
        - Execute dropping of table titled ‘_BACKUP’
    - Create new backup tables
        - Execute renaming old table to ‘_BACKUP’
- Add new data
    - Create new table
        - To create new tables in Python, first create table in SSMS. Remember to refresh connection. To get script for Python, right click on table, select ‘Script table as’ → ‘Create to’ → ‘new query editor window’
            - Copy and paste into Python. Remove all ‘go’ from script.
        - Can also be created by viewing data and writing script based on columns.
    - Insert dataframe into new table
        - Establish a connection with an engine, software that connects to relational database.
        - Send dataframe to SSMS using to_sql

Folder Breakdown

Each folder in main directory will have the following:

- Backups
  - Here the data from the previous update will be saved until the next update.
- Notebooks
  - The Jupyter Notebooks upon which the Python scripts are based are located here.
    - Note: They may not match their respective Python script. If this is case, open ‘command prompt’, navigate to the notebook folder, and run ‘jupyter nbconvert --to script [notebook].ipynb’
      - If you want to convert all the notebooks to .py scripts use asterisk (*) instead of the notebook name
- Scripts
  - Python Scripts that the .bat file runs are located here. After converting notebooks to .py files, move .py files here.
- Updates
  - The data that was pulled from the latest update will be here, waiting to be uploaded to the database.

Some folders will have a Data folder for data in .csv form that could not be pulled using web address.

All folders contain or will contain Windows Batch Files (.bat) to simplify and expedite update process.

Updating Data

Each category folder will have a .bat file. This file can be used to update the respective categories data with two clicks.

- In order to update the data, you will have to clone the repository onto your machine. (If you already have the repository downloaded, skip to the next bullet)
  - To do this:
    - On GitHub, navigate to the main repository, NCDataDashboard.
      - Click the green 'Clone or download' box.
        - Copy the link.
    - Open VSCode
      - Click the 'View' menu on the top bar
        - Select Command Palette
          - You can use 'Ctrl+Shift+P' as a shortcut for this step
      - Choose or type Git: Clone and paste the copied link.
        - Hit Enter
        - Choose a place to save the repository
          - Desktop should do
- Once the repository has been downloaded, to update:
  - Open Command Prompt
    - Type 'cmd' into Windows search bar
    - Navigate to path of repository using 'cd' → 'Enter' to move forward and 'cd..' → 'Enter' to move back
      - Example:
        C:\Users\username\cd Desktop 'enter'
        C:\Users\username\Desktop cd NCDataDashboard 'enter'
    - When in folder, type or tab to update.bat and press 'Enter'
      - When testing, use update_test.bat
    - Let the files update.
      - update.bat will also push updates to GitHub as another form of backup
      - When that finishes, the database, Excel, and GitHub repository should all be updated. Double check to see the latest updates by the repository on GitHub.com.

Notes

In Dashboard testing server, Stein, Jupyter may operate different from local machines. To open Jupyter notebooks:

```
python -m notebook
```

If you do not have git and VSCode (or the like), there is a way to use these scripts.

- Instead of cloning repo, it can be downloaded as zip file from GitHub.
    - Go to NCDataDashboard on GitHub.com
    - Click ‘Clone or Download’
    - Click ‘Download as Zip’
        - Download should start
        - When finished, extract files to known location (e.g. Desktop)
        - To run, view Updating Data section above
            - Update_test.bat has all git commands removed to increase usability

A Unix version of the code can be found on my main account in the ‘DataDashboard_Unix’ repository (https://github.com/nathayoung/DataDashboard_Unix)