CANDATAPY
A System for download of Cansim Data
Alpha Version 0.2

Harold Henson

1 Introduction

Analysts working in large organizations frequently must maintain databases that support libraries of analytical tables. Challenges frequently occur in keeping these databases up to date and coping with changes in the structure of the data. Historically, teams of analysts have built substantial amounts of analytical capital with older proprietary software such as AREMOS. The collapse of budgets and the advent free data systems such as PYTHON have indicated the potential for change. CAMDATAPY is a Python framework designed to allow the easy automation of the management of data provided by the Canadian government through the Cansim system.

This first version of CANDATAPY was written with a very definite application in mind. It has been documented with the full knowledge that the market of potential is limited. However, if it proves to be worthwhile, it should be possible to replicate and adapt to other situations.

The documentation starts off with a motivation of the design of the system. It is based on the authour's experience in both working on and managing teams of analysts that center their work around large systems of databases. For this reason, this documentation provides a longer section on motivation than is usual. It is hoped that the discussion of rationale for the design of this system will help a potential user better decide whether the system is suitable for their application.

After the section on motivation, there is a section that provides an overview of the use of the system. On seeing the role played by the major pieces of the system the technical details should make sense. As well, the individual components are generously documented.

2 Overview

Many analysts work in situations where they must provide complex up to date analysis based on tables that have been generated on the fly. This implies that a piece of analysis with tables and graphs must be completed in periods of time as short of a few hours. This is only possible if the analyst can safely assume that the data in the system is correct. In turn this is only possible when the software that maintains the system is stable.

CansimPY is designed to capture the data as it becomes available and incorporate it into a Pandas database. An overview of the workflow would be:

- Notification of an update to a Stats Can Matrix
- Analyst downloads the matrix from Statistic Canada web site using the sites interactive software
  - Attempts to replicate previous effort to the extent possible
- Matrix update command incorporates the new data into the central Panda database

  - Programs can be run from this central database to update various BI products
  - Previous version of database saved and summaries of the changes are produced

The system will work in identical fashion if it is used for a single user or for a group of analysts who use a common pool of data and need to be consistent. However, the single user can use the same system and skip some of the steps.

In software terms, this should be thought of as a framework. It consists of a few modules that provide an environment for the easy maintenance of a large database of Cansim data. Analysts should be able to maintain a Pandas database by downloading the Cansim data, and rerunning Python programs.

2.1 Motivation for Design

Many analyst are primarily subject matter experts, yet data can be the lifeblood of their work. Frequently, they work in teams that are not sufficiently large enough to justify a pure database expert. In such cases, the teams are frequently composed of a variety of individuals with varying levels of expertise in database management. In such a circumstance, the responsibility for maintaining the integrity of the collective data typically rests with one or two individuals who have the motivation and expertise.

This system is based in Python, on the assumption that although it is a programming language, it is easy enough for senior analysts to use. For organizations where this is not true, then obviously another solution is necessary. Still, it is anticipated that the use of Python by organizations who need to maintain data systems to support analysis relevant to an organization will only increase, at the expense of cost proprietary system that are not any easy to use when there is a significant amount data wrangling required to get the data in a useable format.

2.2 Target Users

The target users of the system are two-fold. First there are those who must execute the manual downloads of the Cansim data, then run the Python programs to incorporate the data into the central database. Their understanding of the system can be limited. It is only necessary for them to follow the examples in the tutorials to perform the necessary functions.

The bulk of this document targets the second class of users who assume overall responsibility for the data in their team. This class of user will first install the system on a common drive for their team. Then modify the system as requirements changes. This will include:

- Augmenting the list of Matrices from which data is downloaded
- Dropping data no longer needed from existing system

- Changing the structure of directories for which the data is kept

A serious attempt is also being made to ensure that the code is well documented, so as to complement this documentation as well.

2.3 Major Components

The system is loaded in a Python module known as CandataPY. Once this is installed on the system, all team members should be able to use the command import CandataPY from their account. This system will contain three major components. They will be briefly summarized in this overview section, but the section on System Components will document the system in considerable detail.

2.3.1 CanDataPY

This is the module that will be imported by the user to define their session. There are three important methods. A setup program is available to install the directory structure and files to allow the system to run. It should only be run once.

2.3.2 CanDataMatrix

A substantial portion of setting up the system will occur at this stage. A separate class for each Stats Can Matrix will be prepared. These classes will inherit from the generic StatCanMatrix class. The manipulations specialized to this data will be included in the inherited class. For example if industry specific data needs to be aggregated in a certain fashion, then expressed as shares, it could be done at this stage.

2.3.3 TimeVar.py

This object contains the finest level of detail. When a matrix is updated each one of these objects will be invoked from TSlist. It will read the data, and then compare it with what is already in the Pandas database. If the data is not within an acceptable level of tolerance, the version on the Pandas database is not changed and a warning is issued.

3 Use

This module should be thought of as a framework built in Python. Once the startup program has been run, the analysts will created a program for each matrix that can be run to add the data from the matrix to a central database. These programs can be saved and run on an as needed basis.

3.1 Run startup program

The first step is to establish a directory in the network shared by all members of the team. Analysts who are maintaining the data base should have full access to this directory and its sub-directories. All possible users should have at least read access to the directory where the final Pandas database should reside.

The second system is to install the CansimPY module on the "System", so that all analysts maintaining the database will be able to import the module when they run a Python program. As this is very site specific, not much will be said in this document. However, it should be no different than installing any other shared module for team use.

After that go into Python, and set the working directory to the target directory. Then from the Python consule enter the command CansimPY.setupdirs(). This will setup all the directories. In general it is advised not to change any of the defaults.

For the subsequent discussion the root directory is refered to as Cendir. Subdirectory will be designated with periods to avoid confusion with the direction of the slash.

3.2 Download the data

It is recommended that the analyst download the data from Statistics Canada down to a local drive. Then they can go into their spreadsheet viewer and look at the data to quickly verify by means of comparison of the previous download of that matrix, which resides in Cendir.rawdownloads. The bulk of the possible problems can be identified with this visual inspection. As confidence in the system grows, the analyst may skip this step and rely on the automated procedures.

If the analyst is confident that he has downloaded the data correctly then the freshly downloaded data should be copied down into Candir.rawdownloads for further processing. The analysts should not be overly concerned that they may have overwritten good data as Cendir.archives contains the downloaded data that is in use with the current Pandas database.

3.3 Load Matrix of data into Central Database

There will be a CandataPy object underlying every matrix. The analysts will simply execute the update method for that matrix. This should fit into the workflow of the group in a natural way as all the data from any given matrix will typically come from the same source and be updated at the same time.

Most of the time, things will work with very little effort. However, the system may produce warnings that need to be addressed. At this point in time a decision may have to be made about whether the data needs to be rethought. Many times the warnings can be ignored but other times adjustments must be made. It is important to note that these decisions can only be made by a subject matter expert, as it is not a technical issue.

3.4 Keep data up to date

Once a matrix of data has been added to the central database, the source code will be saved in the archive directory. At a future date, this code can be rerun without significant modification so as to update the database.

3.5 export data

It is certainly possible to build reports off this system. In fact, a demo of this system will be built there the database drives some Django web tables. However, it is recognized that much analysis will occur with the more specialized data that exists. As such, all of the export capacity that exists for a Pandas database still exists.

4 System Components

This section is written as a reference and will be frequently updated. Its goal is support the day to day use, rather than explain the system. This section may form a separate document if it becomes disproportionately large relative to the rest of the document.

4.1 environment

It is assumed that all possible user's of the data will have access to a common directory. Through out this document, this common directory will be referred to as cendir, although the actual name will be determined by the local environment. A period will be used in place of a slash when discussing sub-directories, as the direction of the slash may vary with the operating system.

In this directory a file central_data.h5. This file type supports a write once read many times usage. It is thus expected that all potential data users will have access to this file. Those few who will maintain the database should have read and write access. When the installation is being executed by the startup program, it should be possible to create subdirectories in cendir.

The system is written to favour the recent versions of Python 3. As Pandas is sometimes difficult to install on some environments, Anaconda is often the simplest solution as Pandas is part of the suite. There may be prompts for other packages, but they can be resolved with PIP.

4.2 modules

The three basic modules were introduced in the Overview section.

4.2.1 CanDataPYSession.py

This module has two roles. First, it sets up the directory structure in Cendir. There is a startup.py in this module that is run once. Second it provides a

parent class that is inherited by StatsCanMatrix. This parent class maintains information common to all updates of the matrices.

4.2.2 StatsCanMatrix.py

The data from Statistics Canada web site will be downloaded in groups referred to as as matrices. In general, individuals will download a matrix that may or may not contain every possible Timeseries from the matrix. Still, the Cansim matrix constitutes a fundamental organizing principle around the data. As such, this system has a corresponding object that stores the data. The information in this matrix is persistent, which allows it to be updated on a regular basis. The StatCanMatrix object will be the parent of the objects that are created for each Stats Can matrix.

The definition of each new class for each download to occur will contain information particular to that matrix. An example is provided in the sample listing at the end of this document. In this listing the user provides four pieces of key information. These are sufficient to allow the loading of the downloaded CSV file into the Panda file.

- Update - This method will attempt to load the downloaded csv file. It will return True if the data is successfully downloaded and converted to the Pandas format. The results of this process will be stored in a dictionary update_info.
- __str__ This is overridden to allow the quick view of the update_info dictionary

4.2.3 timevar.py

The Panda's Series is the fundamental building block of the system. The module timevar.py defines a class, which builds up the data to create a Panda Series. When all of the data has been read in, this data is saved to a dataframe that it has been saved to. This process is repeated until all the desired data for the matrix has been added to thematrix.thepandas.

This class exploits the fact that some aspects of the series are the same for all the series on a Matrix. As such, there is one initialization program for all of the parameters, and there is a second for just the variables that may change. Detailed documentation is embedded in the source code itself, however and overview of the key methods follows:

- __init__ - The frequency is established here. As well, a link is formed to the the larger session with the thematrix. Note that a method is used to initialize the various values, which in turn calls two different routines.
- setdate(thetoken) - This method will take a token and parse out it according to the frequency that was established in __init__. The datstr is created so that it can be used by save to create the series

- setvalue(thetoken) - This method converts thetoken into a float and appends it to the values list. If the thetoken cannot be converted to a real number, it will be converted to numpy.nan if it is '..','...', or 'x'. Any other value it is assumed that there is some kind of processing error and the entire series is not used. The name of the unused series is written out to the error log. See Appendix for some details on this issue.

- save - When the last observation has been read in, this method will be called to add the time series to the dataframe. For this method to work properly:

  - the beginning period needs to have been established
  - the number of observations should be known
  - there should be a vector the length of the number of observations
  - the name of the variable should have been established

5 Future Directions

This version contains a very minimal capacity. While a reasonable as a first step, there is still much than be done to fully automate the process of automating the production of Business Intelligence with Cansim data. As it stands, there are three major areas of potential development:

- Automate Download - Currently the assumption is that analysts will use the Stats Can interactive environment to download in the Matrix. Hopefully, the updates will follow the same pattern every time the database is updated. However, this process is inefficient and error-prone in comparison to a possible automated process.

  It is technically possible to automate this process. However, it is important to note that Statistics Canada's position on this has been in flux in recent history. Still, a utility to replicate the download process may be very useful at some point in the future.

- Quality Control of Data - It is impossible to manually verify all the after each download. However, if comparisons could be done with a version that had been manually verified, then many possible problems could be captured. A section of this is included in Version 0.1 of this data, but has been omitted for now.

- Automated tables in Django - It would also be very possible to automated the production of reports in Django for Intranet/Internet sites. Not only would the tables be useful in themselves, they would also provide an opportunity to blend narrative data that helps understand the quantitative data coming from Statistics Canada.

6 Annex

6.1 Treatment of Invalid Data

Data downloaded from Cansim should be numeric. However, if there are problems Statistics Canada will supply text in place of the numbers. To date, three text items have been identified that flag reasons that an numeric value of an observation is not available:

- 'x' - Data suppressed to meet confidentiality requirements.
- '..' - Data not available for the reference period
- '...' - figures on applicable In all three of these cases, the Pandas value for NA is inserted in the observation. Note that this usage of is defined in page 140 of McKinney.

6.2 Code Samples

The box gives a working example that loads Matrix282001 into Pandas database.

Listing 1: A matrix inherited class example

```
class Matrix282_0001(StatCanMatrix):
    def__init__(self, thefile, thename="282"):
        super(Matrix282_0001, self).__init__(thefile, 7000,
              matname=thename)
        # these are valid keys
        self.setcollist(['date', 'NA', 'NA', 'gender',
              'NA', 'VName', 'NA', 'datum'])
    def upload(self):
        super(Matrix282_0001, self).upload('obs_by_row',
              'M', 'first5')
# create an instance of it
this282 = Matrix282\_0001("02820001-eng.csv",2820001)
# to the actual upload
this282.upload()
```