Working with Large Files & Data Ingestion Pipeline

Alireza Ehiaei

19-July-2021

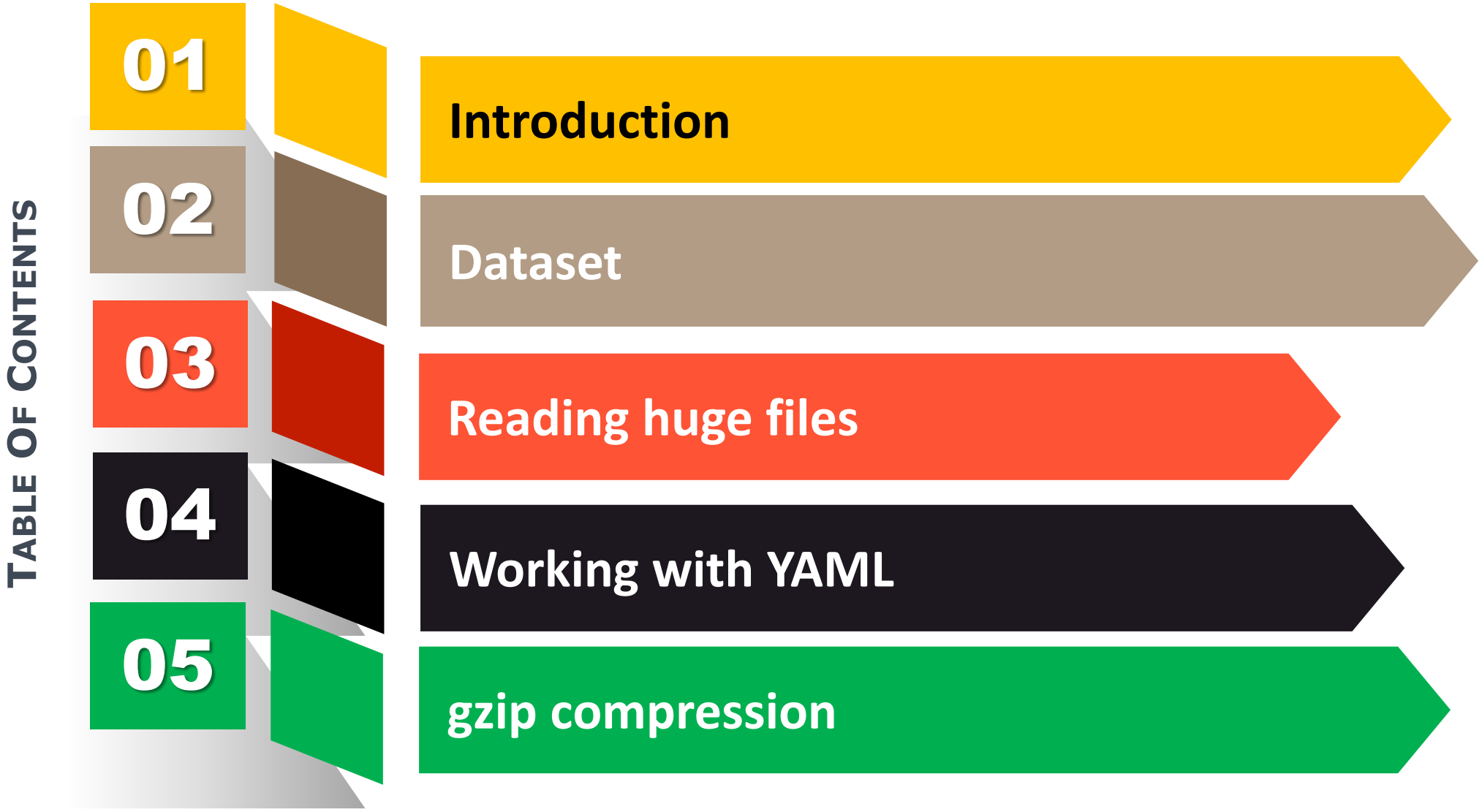
Table Of Contents
01
Introduction
02
Dataset
03
Reading huge files
04
Working with YAML
05
gzip compression

Introduction

Loading Big Data or trying to read very big (more than 1 GB) CSV files usually arise Memory Errors while. There are some ways how to handle it in Python environment.

In this project different ways of reading of huge data files are reviewed and by creating a pipeline some basic data manipulation is provided.

Dataset

- Data set includes two csv files in Stata format, both almost 1.4 GB totally 2.9 GB.
- The data source is:

  https://dataverse.harvard.edu/dataverse/atlas/?q=country_hsproduct6digit_year

  https://dataverse.harvard.edu/dataverse/atlas/?q=country_partner_sitcproduct2digit_year
- Data and code are uploaded at:

  https://github.com/Alireza-Ehiaei/Data_Sciences/tree/main/Reading_huge_data

Dataset

- Data downloaded from the Atlas of Economic Complexity that maintains trade data in multiple international classification systems.
- Categorizes approximately 5,000 goods
- Covers years from 1995–2018
- Raw data on trade in goods is provided by United Nations Statistical Division (COMTRADE). The data is then cleaned by Growth Lab researchers using the Bustos-Yildirim Method which uses bilateral trade flows to account for inconsistent reporting and provides more reliable accounting.
- In addition to trade in goods, the data additionally contains unilateral data on services trade provided by the International Monetary Fund (IMF) and acquired through the World Development Indicators (WDI) of The World Bank.

Reading huge files

Using chunk of data:

Creating an iterator, which reads the metadata attached to the file that doesn't read the data itself. Then read in just a chunk of the data at a time.

```
)3]:
1 %%time
2 with open(r"C:\Users\IMBS\Downloads\programming\Atlas-Harward\country_hs
3     n_rows = len(file.readlines())
4
5 print ('Exact number of rows: {}'.format(n_rows))
```

```
Exact number of rows: 2279873
Wall time: 20.6 s
```

```
)4]:
1 itr = pd.read_stata(r"C:\Users\IMBS\Downloads\programming\Atlas-Harward\
```

```
)5]:
1 # Now it's possible to read in just a chunk of the data at a time.
2 auto = itr.get_chunk(5)
3 auto
```

Reading huge files

The first rows of the second file above (country_hsproduct6digit_year):

	location_id	product_id	year	export_value	import_value	hs_eci	hs_coi	location_code	hs_product_code
0	0	5000	1995	18008.0	7199.0	-0.468138	-0.696617	ABW	010111
1	0	5000	1996	0.0	3020.0	-0.663710	-0.704456	ABW	010111
2	0	5000	1997	NaN	NaN	-1.194294	-0.818992	ABW	010111
3	0	5000	1998	NaN	NaN	0.199708	-0.704800	ABW	010111
4	0	5000	1999	14510.0	46679.0	-0.083034	-0.801171	ABW	010111

Reading huge files

We can also easily loop over the data like so:

```
itr = pd.read_stata(r"C:\Users\IMBS\Downloads\programming\Atlas-Harward\country_hs

col4 = pd.DataFrame()
for df in itr:
    col4 = col4.append(df)
    if len(col4.index)==30:
        break
col4
```

	location_id	product_id	year	export_value	import_value	hs_eci	hs_coi	location_code	hs_product_code
0	0	5000	1995	18008.0	7199.0	-0.468138	-0.696617	ABW	010111
1	0	5000	1996	0.0	3020.0	-0.663710	-0.704456	ABW	010111
2	0	5000	1997	NaN	NaN	-1.194294	-0.818992	ABW	010111
3	0	5000	1998	NaN	NaN	0.199708	-0.704800	ABW	010111
4	0	5000	1999	14510.0	46679.0	-0.083034	-0.801171	ABW	010111
5	0	5000	2000	0.0	18778.0	0.349085	-0.777499	ABW	010111
6	0	5000	2001	10544.0	19844.0	-0.104944	-0.835741	ABW	010111

Reading huge files

We can also easily loop over the data like so:

```
itr = pd.read_stata(r"C:\Users\IMBS\Downloads\programming\Atlas-Harward\country_hs

col4 = pd.DataFrame()
for df in itr:
    col4 = col4.append(df)
    if len(col4.index)==30:
        break
col4
```

	location_id	product_id	year	export_value	import_value	hs_eci	hs_coi	location_code	hs_product_code
0	0	5000	1995	18008.0	7199.0	-0.468138	-0.696617	ABW	010111
1	0	5000	1996	0.0	3020.0	-0.663710	-0.704456	ABW	010111
2	0	5000	1997	NaN	NaN	-1.194294	-0.818992	ABW	010111
3	0	5000	1998	NaN	NaN	0.199708	-0.704800	ABW	010111
4	0	5000	1999	14510.0	46679.0	-0.083034	-0.801171	ABW	010111
5	0	5000	2000	0.0	18778.0	0.349085	-0.777499	ABW	010111
6	0	5000	2001	10544.0	19844.0	-0.104944	-0.835741	ABW	010111

Reading huge files

Getting some data at which the data were last saved without importing the file

```
[98]: 1 itr.nobs
: 30385560

[99]: 1 itr.nvar
: 9

[00]: 1 itr.time_stamp
: '18 Jun 2020 16:35'

[01]: 1 itr.varlist
: ['location_id',
   'product_id',
   'year',
   'export_value',
   'import_value',
   'hs_eci',
   'hs_coi',
```

Reading huge files

the first column takes up 4 bytes

for each row, ...

```
2  itr.col_sizes
```

```
[4, 4, 4, 8, 8, 4, 4, 3, 11]
```

Reading huge files

Converting first 1000 rows of data frame to csv file:

```
import pandas as pd
itr = pd.read_stata(r"C:\Users\IMBS\Downloads\programming\Atlas-Harward\country_hspro

dff = pd.DataFrame()
for df in itr:
    dff = dff.append(df)
    if len(dff.index)==1000: #Takes less time as compared to reading the entire file
        break
```

```
dff.to_csv("df1.csv") #the file will be saved in the path of stata file if it is dow
```

Reading huge files

211]:
```
%%time
df_pd = pd.read_csv(r"C:\Users\IMBS\Downloads\programming\Atlas-Harward\df1.csv")
df_pd
```

```
Wall time: 18 ms
```

:

	Unnamed: 0	location_id	product_id	year	export_value	import_value	hs_eci	hs_coi	location_code	hs_product_
0	0	0	5000	1995	18008.0	7199.0	-0.468138	-0.696617	ABW	1
1	1	0	5000	1996	0.0	3020.0	-0.663710	-0.704456	ABW	1
2	2	0	5000	1997	NaN	NaN	-1.194294	-0.818992	ABW	1
3	3	0	5000	1998	NaN	NaN	0.199708	-0.704800	ABW	1
4	4	0	5000	1999	14510.0	46679.0	-0.083034	-0.801171	ABW	1
...	...	...	...	...	...	...	...	...	...	
995	995	41	5000	2006	1124968.0	2923882.0	2.052127	0.536750	CHE	1

Reading huge files

Using dask

2]:
```
1 import dask.dataframe as dd
```

3]:
```
1 df_dask = dd.read_csv(r"C:\Users\IMBS\Downloads\programming\Atlas-Harward\df1.csv")
```

5]:
```
1 %%time
2 df_dask.compute()
```

```
Wall time: 19.9 ms
```

	Unnamed: 0	location_id	product_id	year	export_value	import_value	hs_eci	hs_coi	location_code	hs_product_
0	0	0	5000	1995	18008.0	7199.0	-0.468138	-0.696617	ABW	1
1	1	0	5000	1996	0.0	3020.0	-0.663710	-0.704456	ABW	1
2	2	0	5000	1997	NaN	NaN	-1.194294	-0.818992	ABW	1
3	3	0	5000	1998	NaN	NaN	0.199708	-0.704800	ABW	1
4	4	0	5000	1999	14510.0	46679.0	-0.083034	-0.801171	ABW	1

Reading huge files

Using pickle

Save it as pickle for faster loading

```
.8]:  1  %%time
      2  df_all = pd.read_csv(r"C:\Users\IMBS\Downloads\programming\At
```

```
Wall time: 3.99 ms
```

```
:0]:  1  # Save as Pickle
      2  df_all.to_pickle("newdataset.pkl")
```

Reload it and utilize it

```
:1]:  1  %%time
      2  new_df = pd.read_pickle("newdataset.pkl")
```

```
Wall time: 3.99 ms
```

YAML Configuration

YAML stands for YAML Ain’t a Markup Language. It is a recently introduced data serialization format and is very comfortable for human reading and writing.

It is often used for configuration files, but can also be used for data exchange. The most used python YAML parser is PyYAML library.

features of YAML[1]:

- You can use comments in YAML files
- You can store multiple documents in one YAML file, with the --- separator
- It’s easy to read for humans
- It’s easy to parse for computers

1 - https://python.land/data-processing/python-yaml

YAML Configuration

The YAML parser returns a regular Python object that best fits the data. In this case, it's a Python dictionary. This means all the regular dictionary features can be used, like using get() with a default value.

creating a yaml file in the path directory:

```
1 import yaml
```

```
1 %%writefile file2.yaml
2 file_type: csv
3 file_name: df1
4 skip_leading_rows: 1
5 columns:
6     ['location_id', 'product_id', 'year', 'export_value',
7        'import_value', 'hs_eci', 'hs_coi', 'location_code', 'hs_product_code']
8
```

YAML Configuration

Creating a pipeline using yaml file to fulfil some preprocesses:

```
%%writefile modification.py
import logging
import os
import subprocess
import yaml
import pandas as pd
import datetime
import gc
import re

# File Reading #
def read_config_file(filepath):
    with open(filepath, 'r') as stream:
        try:
            return yaml.safe_load(stream)
        except yaml.YAMLError as exc:
            logging.error(exc)

def col_header_val(df,table_config):
    '''
    replace whitespaces in the column
    and standardized column names
```

YAML Configuration

Reading yaml file:

```
[ ]:  1  import modification as mod
      2  config_data = mod.read_config_file("file2.yaml")
```

```
[ ]:  1  config_data
```

```
{'file_type': 'csv',
 'file_name': 'df1',
 'skip_leading_rows': 1,
 'columns': ['location_id',
  'product_id',
  'year',
  'export_value',
  'import_value',
  'hs_eci',
  'hs_coi',
  'location_code',
  'hs_product_code']}
```

YAML Configuration

read the csv file using yaml config file:

```
3  import pandas as pd
4
5  file_type = config_data['file_type']
6  source_file = "./" + config_data['file_name'] + f'.{file_type}'
7  #print("",source_file)
8  df = pd.read_csv(source_file)
9  df.head()
```

	Unnamed: 0	location_id	product_id	year	export_value	import_value	hs_eci	hs_coi	location_code	hs_product_co
0	0	0	5000	1995	18008.0	7199.0	-0.468138	-0.696617	ABW	101
1	1	0	5000	1996	0.0	3020.0	-0.663710	-0.704456	ABW	101
2	2	0	5000	1997	NaN	NaN	-1.194294	-0.818992	ABW	101
3	3	0	5000	1998	NaN	NaN	0.199708	-0.704800	ABW	101
4	4	0	5000	1999	14510.0	46679.0	-0.083034	-0.801171	ABW	101

YAML Configuration

yaml results are more readble:

```
print("columns of files are:" ,df.columns)
print("columns of YAML are:" ,config_data['columns'])
```

```
columns of files are: Index(['Unnamed: 0', 'location_id', 'product_id', 'year', 'export_value
',
       'import_value', 'hs_eci', 'hs_coi', 'location_code', 'hs_product_code'],
      dtype='object')
columns of YAML are: ['location_id', 'product_id', 'year', 'export_value', 'import_value', 'h
s_eci', 'hs_coi', 'location_code', 'hs_product_code']
```

YAML Configuration

Validating the header of the file:

```
2
3
4 util.col_header_val(df,config_data)
```

```
column name and column length validation failed
Following File columns are not in the YAML file ['unnamed_0']
Following YAML columns are not in the file uploaded []
```

: 0

GNU zip (gzip) compression

Getting some information from file with txt.gz format:

```
import gzip
import shutil
import csv
with open("df1.txt", 'rb') as inpute:

    with gzip.open("df1.txt.gz", 'wb') as output:
        shutil.copyfileobj(inpute,output)
```

GNU zip (gzip) compression

Compress as a text gzip format:

1]:

```
1  import os
2
3  myfile = r"C:\Users\IMBS\Downloads\programming\Atlas-Harward\df1.txt.gz"
4
5  with gzip.open(myfile, 'rb') as f:
6          for i, l in enumerate(f):
7              pass
8          print("df1.txt.gz contain {0} lines".format(i + 1, myfile))
9
10         print("file size of df1.txt.gz is", os.stat(r"C:\Users\IMBS\Downloads
```

```
df1.txt.gz contain 1001 lines
file size of df1.txt.gz is 20534
```

Thank You