Lecture 2 (part 1): Loading data to Python

Advanced Business Analytics (CIS442D/85)

Simon Business School

1/11/2017

Announcements

- Homework 1 - submission extended to Thursday, 1/12 at 23:55
- Homework 2 due Tuesday, 1/17 at 23:55
- Change in office hours (location and time in the syllabus)
- Form teams of 3-4 students and update Solomon (TA) by email

Lecture Outline

Goal: explore tools and techniques for loading and storing data in Python

1. csv
2. json and web-services
3. sql
4. xml
5. Web-scraping
6. Summary

CSV

- Comma-Separated Values
- Text files that represent tables

Language	Wiki	Articles	Pages	Edits
English	en	5236357	40155905	847883223
Swedish	sv	3445556	6910072	36854915
Cebuano	ceb	2885091	5354566	10393518
German	de	1976573	5699722	162755765
Dutch	nl	1873902	3637030	48492474

- Content of data1.csv
- Columns are separated by commas, rows by newlines (lines breaks)

```
Language,Wiki,Articles,Pages,Edits
English,en,5236357,40155905,847883223
Swedish,sv,3445556,6910072,36854915
Cebuano,ceb,2885091,5354566,10393518
German,de,1976573,5699722,162755765
Dutch,nl,1873902,3637030,48492474
```

- Often the first row contains the column titles (header)

CSV - cont.

- Inspecting files: !pwd, !ls, !head
- Load data using pandas to DataFrame using read_csv()
  - df = pd.read_csv('data1.csv')
  - Missing header:

    pd.read_csv("data2.csv", header=None, names=['Language','Wiki','Articles','Pages','Edits'])
  - Skip rows, custom column separator, custom number specification, missing values, compressed files, read files in chunks, ...
- Export DataFrame to file using to_csv():
  - df.to_csv('data_out.csv')
  - Subset of columns, indices, separator, ...
- Read csv record into a list using Python csv library

Processing large files

- Compressed files (read_csv supports the formats gzip, bz2, zip, and xz)
- Parameter 'nrows' (useful for debugging)
- Parameter 'chunksize' (returns a collection of dataframes)
- Iterate manually with Python's built-in packages
  - line-by-line text reading
  - line parsing with the csv package

Exercise 1

See "Lecture - Loading data in Python (csv).ipynb"

Exercise 2

See ”Lecture - Loading data in Python (csv).ipynb”

json: JavaScript Object Notation

- Widely used data format
- Very common for transferring data over the Internet (Twitter, Google maps, OpenCorporates)
- Example of json string:

```
rec = """
{
"name": "Manuel Calevera",
"age": 40,
"friends":["Mercedes", "Glottis", "Olivia", "Sal"],
"height":null,
"clues": {"a":1,"b":2}
}"""
```

- Specifies data using numbers, booleans, strings, lists, dictionaries (called objects), and the reserved word null
- Very similar to Python (keys must be characters)

Reading/writing strings and files

- Parse json string:

```
obj = json.loads(rec)
print(obj['friends'])
print(obj['clues']['b'])
```

- Convert object to json string

```
s = json.dumps(obj)
```

- Export object to json file

```
json.dump(obj, open("json_example.json","w"))
```

- Parse json file

```
json.load(open("json_example.json","r"))
```

- json package documentation

Reading/writing DataFrames

- pandas provides simple interface for working with json files
- Exporting DataFrames to json files (to_json)

```
df.to_json("output.json", orient="split")
```

- Support variety of output modes: split, records, index, columns, values
- Load json file into Dataframe (read_json)

```
pd.read_json("data.json", orient="values")
```

- json allows very general representation of data (vs. tabular)
- May use json.load and json.loads for files that do not fit the above structure use

Web-services

- Web interface to servers (e.g., databases)
- Example: OpenCorporates - open database on world-wide companies
- Search results
- API (application programming interface): OpenCorporates

Exercise 3

Print the names of the companies returned by the search "BARCLAYS BANK"

Databases

- Database Management System: software designed to store, retrieve and analyze data
  - Common source of data (e.g., extracted from an ERP system)
  - Can be used to process “medium-size data”
- Relational databases
  - A common model for storing data
  - Data is structured in relations (tables with similar type across columns)
  - Focus on SQLite and its built-in Python interface sqlite3

SQLite

- Relational DBMS
- The most widely used database engine (web-browsers, OS)
- Contained in a single disk file (embedded)
- Cross-platform, open-source, many interfaces
- Typical usage
  - Create and Drop tables
  - Insert records into tables
  - Update and Delete records
  - Query: extract data from database
- Use sqlite3 and pandas

Python sqlite3 library and pandas

- Connect to database: `sqlite3.connect('db.sqlite')`
- Query tables using pandas: `pd.read_sql_query("SELECT * from sqlite_master", con)`
  - "Master table" lists the tables in the database
  - Alternatively: Database manager or SQLite command line interface
- Create table:

```
con.execute("CREATE TABLE tbl(wikipedia TEXT, topic TEXT, year INTEGER, month INTEGER, pageviews INTEGER);")
con.commit()
```

- Insert records: `con.execute("INSERT INTO tbl VALUES %s"%str(tuple(rec)))`
- Query tables: filter, aggregate, sort, create table from query, ...
- Resources: sqlite3 library, SQLite, Wikipedia, tutorials: [1], [2], [3]

XML (Extensible Markup Language)

- Data format
- Supports hierarchical data structures
- Standard, popular, and human and machine-readable
- Example: (source)

```
<data>
<country name="Liechtenstein">
<rank>1</rank>
<year>2008</year>
<gdppc>141100</gdppc>
<neighbor name="Austria" direction="E"/>
<neighbor name="Switzerland" direction="W'/>
</country>
<country name="Singapore">
<rank>4</rank>
<year>2011</year>
<gdppc>59900</gdppc>
<neighbor name="Malaysia" direction="N"/>
</country>
</data>
```

XML (Extensible Markup Language) - cont.

```
<data>
<country name=”Liechtenstein”>
<rank>1</rank>
<year>2008</year>
<gdppc>141100</gdppc>
<neighbor name=”Austria” direction=”E” />
<neighbor name=”Switzerland” direction=”W’ />
</country>
</data>
```

- Elements: data, country, year, ...
  - Element type is called ”tag” (e.g., data, country, ...)
  - Root element
  - May contain other elements (list of children)
  - Attributes (dictionary)
  - Text
  - (optional tail)
- XML documents are tree of elements
- Parsing: convert text → list of lists and dictionaries (conceptually)

XML (Extensible Markup Language) - example

```
from lxml import etree
f = open('xml_example.xml','r')
tree = etree.parse(f)
f.close()

root = tree.getroot()
root.getchildren()
root.tag
root[0].tag
root[0].attrib

for e in root.getiterator(): # iterate over all nodes
print(e.tag)

root.xpath('.//year') # list of descendants from type (tag) year
```

Exercise 4

Using the file ”xml_example.xml”, construct a DataFrame that presents the values of year and gdppc per each country (an example of a row in the table is: [Panama ,2011 ,13600]).

Web Data Extraction (web-scraping)

- A wealth of data online, but most often not nicely organized
- Many Python packages for downloading and parsing HTML files (lxml, BeautifulSoup, Python element tree, urllib) and Open-source and commercial tools)
- Simple example using requests for downloading, and lxml for parsing
- HTML is a special type of xml document (open jupyter)

Exercise 5

See the .ipynb file

Web Data Extraction - cont.

- Legal issues
- Additional reading
  - Chapter 6 in Python for Data Analysis
  - HTML Scraping
  - Automate the Boring Stuff with Python

Summary

- Loading data to and from the Python environment
- Discussed common data sources: csv, json, sql, xml, and html
- Managing 'medium-size' data (can fit hard-drive but not memory):
  - compressed files
  - reading files in chunks
  - databases

Optional Reading

- PyTables is an extremely fast and powerful alternative for analyzing "medium-size" data
- pandas-datareader - interface for loading pandas DataFrames from remote databases (Yahoo finance, World bank, OECD, ...)
- NoSQL databases (non-tabular)
  - allow more flexibility and are faster for some operations
  - specific database model should be tailored to the specific system (how data is structured and which operations are commonly used)
  - Examples: MongoDB and HBase
- ODO - conversion between various data formats
- Blaze, SQLAalchemy - database independent querying tools