Intermediate Data Science

Data Loading, Storage, and File Formats

Joanna Bieri DATA201

Important Information

- Email: joanna_bieri@redlands.edu
- Office Hours take place in Duke 209 – Office Hours Schedule
- Class Website
- Syllabus

Data and File Formats

Reading in data and making it accessible to your data science techniques is the first step in data science success. This is called data loading or sometimes parsing. As a data scientists you may not have complete control over the format of your data when you first start a project. So it is part of your job to find a way to load it and parse it.

Here are some important data types:

Data and File Formats

- **CSV (Comma-Separated Values)**

CSV is a simple, text-based format that stores tabular data. Each line in the file represents a data record (a row), and each record consists of one or more fields (columns) separated by commas. The data does not have types and is read in as strings.

Data and File Formats

- **Text - General**

This is a generalization of CSV types where the data could be just written to lines, or stored as a single string. Once this is read in you would need to do a lot of work to parse it and turn it in to data. It is extremely inefficient, but sometimes this is how the data comes to you.

Data and File Formats

- **JSON (JavaScript Object Notation)**

JSON is a text-based format for representing semi-structured data using key-value pairs, similar to Python dictionaries. It's great for data that has a nested or hierarchical structure. It is human readable but less efficient for storing large flat tables.

Data and File Formats

- **Apache Parquet**

Parquet is a binary, columnar storage format. Instead of storing data row-by-row like a CSV, it stores all the values for a single column together. This structure is highly optimized for analytical queries. This format can make it really fast to read in subsets of the data and allows for compression of the data. It is not human readable.

Data and File Formats

- **Pickle**

Pickle is a Python-specific binary format used for serializing and de-serializing Python objects. It can turn almost any Python object (like a list, a dictionary, or even a trained machine learning model) into a stream of bytes that can be saved to a file. It can handle very complex python objects and is extremely easy to use. Can have some compatibility and security issues.

Data and File Formats

- **HDF5 (Hierarchical Data Format 5)**

HDF5 is a high-performance binary format designed to store and organize massive amounts of data. It acts like a file system within a single file, allowing you to store multiple datasets (e.g., arrays, tables) in a hierarchical structure. It is great for large,complex scientific datasets, but has a steep learning curve.

Data and File Formats

- **Geospatial Data**

Geospatial data are typically categorized as either vector (representing features with points, lines, and polygons) or raster (representing features as a grid of pixels or cells). There are many formats for these files. One popular one is GeoJSON - which leverages the JSON file format but allows for vector based data. GeoTIFF is standard for raster data.

Data and File Formats

- **Web APIs (Application Programming Interface) - not a data type but a data access type**

Many websites have public APIs providing data feeds via one of the formats above. If a website has an API , you should use this interface to get data rather than trying to scrape the site.

Data and File Formats

- **Databases - again not a data type**

In many cases data can be stored in a database: SQL server, MySQL, NoSQL Mongo, or Graph Databases. In these cases you *query* the database to get access to subsets of the data. The choice of database is highly dependent on the project needs and scalability.

Pandas Data Loading Functions

Most of this class will focus on using Pandas for our data management. NOTE: if you are a geospatial student there is a sister package called GeoPandas that has very similar functionality to Pandas.

Function	Type	Description	Common File Types
`pd.read_csv()`	Text	Reads comma-separated values into a DataFrame.	`.csv`
`pd.read_table()`	Text	Reads general delimited text files (default delimiter is tab).	`.txt`, `.tsv`
`pd.read_fwf()`	Text	Reads fixed-width formatted text files.	`.txt`
`pd.read_json()`	Text	Reads JSON (JavaScript Object Notation) data.	`.json`
`pd.read_html()`	Text	Parses HTML tables and returns	`.html`

Python Data Loading Functions

- **Read Write**

A standard build in way to read in files. It will read/write text line by line. Does not save variable types and everything is assumed to be a string.

Python Data Loading Functions

```
# ---- WRITE ----
data = [
    "Name,Score",
    "Alice,85",
    "Bob,92",
    "Charlie,78"
]
with open("students.txt", "w") as f:
    for line in data:
        f.write(line + "\n")
```

Python Data Loading Functions

```
# ---- READ ----
with open("students.txt", "r") as f:
    contents = f.readlines()
print("Contents of students.txt:")
for line in contents:
    print(line.strip())
```

Python Data Loading Functions

- **JSON**

Great for reading in or saving dictionaries. It will preserve some data types: dict (keys must be strings), lists/tuple (tuples become lists), string, int, float, boolean, and None. Other data types will not be properly encoded without extra work.

Python Data Loading Functions

```
import json
students = {"Alice": 85, "Bob": 92, "Charlie": 78}
# ---- WRITE ----
with open("students.json", "w") as f:
    json.dump(students, f, indent=4)
    # indent=4 makes it pretty
```

Python Data Loading Functions

```
# ---- READ ----
with open("students.json", "r") as f:
    loaded_students = json.load(f)

print("Data loaded from JSON:")
print(loaded_students)
```

Python Data Loading Functions

- **Pickle**

Great in Python and preserves Python types. However, pickle is Python specific and it is not easy to load .pkl files into other languages.

```
import pickle
students = {
    "Alice": (85, "A"),    # tuple
    "Bob": {"math": 92},   # dict
    "Charlie": [78, 80]    # list
}
# ---- WRITE ----
with open("students.pkl", "wb") as f:
# 'wb' = write binary
    pickle.dump(students, f)
```

Python Data Loading Functions

```
# ---- READ ----
with open("students.pkl", "rb") as f:
# 'rb' = read binary
    loaded_students = pickle.load(f)

print("Original:", students)
print("Loaded:", loaded_students)
print("Types preserved:", type(loaded_students["Alice"]))
```

Optional Arguments in Pandas Data Loading

All of the Pandas functions for Data Loading have optional arguments. These help refine how you load the data, how much data you load, data conversion, and even what to do with bad data. Here are the main categories:

Optional Arguments in Pandas Data Loading

- *Indexing* These arguments help you choose which columns or rows are returned in the DataFrame, and whether or not to get column or index names from the data set.
- *Type inference and data conversion* These arguments help you to converts data types or customize data as you read it in. This might include missing value markers.
- *Date and time parsing* These arguments help you combine date and time information spread over multiple columns into a single column in the result.
- *Iterating* Useful for very large files - you can load in chunks.
- *Unclean data issues* These arguments allow you to skip over header or footer rows, or tell Pandas what to do with comments or numbers that are split by commas.

It can be overwhelming when you see the full list of optional arguments:

See the lecture notes or video for examples

- Reading in CSV files: general, missing data, subsets (chunker)
- Writing to CSV
- Reading and Writing JSON
- Web Scraping
- Pickle and Binary Data
- Microsoft Excel
- Web APIs
- Databases - SQLite
- Kagglehub

Summary

In Data Science you need to be willing and able to interact with LOTS of different data types, file types, and query types. For most of this class we will read the data in directly. DATA 211 Database Management will give you lots more tools for creating and interacting with SQL type databases.

Homework 3

Go to Kaggle Datasets: https://www.kaggle.com/datasets

Find a data set that you are interested in looking at. You are welcome to work together and choose a data set as a group! You should read in this data and do some basic statistics on the data set. Answer the following questions:

Homework 3

1. How many variables and observations?
2. What type of data is contained?
3. Are there any NaNs or weird data types that you can see?
4. Most Kaggle datasets contain some basic stats or visualizations. See if you can recreate some of the plots or data you see on the website.
5. Come up with at least one question of your own that you can answer by analyzing the data.