DataEng S23: Data Validation Activity

High quality data is crucial for any data project. This week you'll gain experience with validating a real data set provided by the Oregon Department of Transportation.

Submit: Make a copy of this document and use it to record your results. Store a PDF copy of the document in your git repository along with any needed code before submitting using the in-class activity submission form.

Initial Discussion Question - Discuss the following question among your working group members at the beginning of the week and place your responses in this space. Or, if you have no such experience with invalid data then indicate this in the space below.

Have you ever worked with a set of data that included errors? Describe the situation, including how you discovered the errors and what you did about them.

Response 1: Yes, I (Max) had to manually fix a csv in a text file for work, about 20 lines changing one value. It would have needed a script if there were more data errors

Response 2: Timofey: I've ran into (sort of an error) with the Trimet data we are using for the project. On the days when no data is available, the API returns [{"data not available for this day"}], which is a problem because it's not a valid JSON object. To avoid attempting dumping such data into a Python dictionary, I would first check the size of the string. If it was relatively short, I would assume that it's an error message and would not parse it.

Response 3: Dan: I personally haven't. But I feel like we should count the dataset that we worked on the previous week. Two datasets that came from the same website (for BeautifulSoup), created problems with our code being robust. I ended up giving up on trying to make the code robust but if I had more time I would've definitely spent it on fixing the code to be able to include both the datasets.

Response 4: Mahshid: During my NLP course, I encountered a dataset with errors while working with Copra, such as mislabeled categories and inconsistent values. I discovered these errors during the exploratory data analysis phase and addressed them using appropriate data cleaning techniques “standardizing labels” & “ filling in missing values”.

The data set for this week is a listing of all Oregon automobile crashes on the Mt. Hood Hwy (Highway 26) during 2019. This data is provided by the Oregon Department of Transportation and is part of a larger data set that is often utilized for studies of roads, traffic and safety.

Here is the available documentation for this data: description of columns, Oregon Crash Data Coding Manual

Data validation is usually an iterative three-step process.

A. Create assertions about the data
B. Write code to evaluate your assertions.
C. Run the code, analyze the results and resolve any validation errors

Repeat this ABC loop as many times as needed to fully validate your data.

A. Create Assertions

Access the crash data, review the associated documentation of the data (ignore the data itself for now). Based on the documentation, create English language assertions for various properties of the data. No need to be exhaustive. Develop one or two assertions in each of the following categories during your first iteration through the ABC process.

1. *existence* assertions. Example: “Every crash occurred on a date”
    a. Every crash has a record type
2. *limit* assertions. Example: “Every crash occurred during year 2019”
    a. The crash hours range from 0-24 only
    b. The crash days range from 0-31 only
3. *intra-record* assertions. Example: “If a crash record has a latitude coordinate then it should also have a longitude coordinate”
    a. If a crash has any date information, then it should have all date info (d/m/y)
    b. The date matches the day of the week
4. Create 2+ *inter-record check* assertions. Example: “Every vehicle listed in the crash data was part of a known crash”
    a. Every crash must have a valid lat/long
    b. Every crash has either a vehicle ID or a serial #
5. Create 2+ *summary* assertions. Example: “There were thousands of crashes but not millions”
    a. There were up to tens of people injured/killed in a crash but not hundreds
    b. There are only 3 record types
6. Create 2+ *statistical distribution assertions*. Example: “crashes are evenly/uniformly distributed throughout the months of the year.”

a. Crashes are clustered in high-risk areas along the US 26 highway
b. Crashes occur at all hours of the day and night

These are just examples. You may use these examples, but you should also create new ones of your own.

B. Validate the Assertions

1. Study the data in an editor or browser. Study it carefully, this data set is non-intuitive!.
2. Write python code to read in the test data. You are free to write your code any way you like, but we suggest that you use pandas' methods for reading csv files into a pandas Dataframe.
3. Write python code to validate each of the assertions that you created in part A. The pandas package eases the task of creating data validation code.
4. If needed, update your assertions or create new assertions based on your analysis of the data.

C. Run Your Code and Analyze the Results

In this space, list any assertion violations that you encountered:

-
-
-
-

For each assertion violation, describe how to resolve the violation. Options might include:

- revise assumptions/assertions
- discard the violating row(s)
- Ignore
- add missing values
- Interpolate
- use defaults
- abandon the project because the data has too many problems and is unusable

No need to write code to resolve the violations at this point, you will do that in step E.

D. Learn and Iterate

The process of validating data usually gives us a better understanding of any data set. What have you learned about the data set that you did not know at the beginning of the current ABC iteration?

Next, iterate through the process again by going back through steps A, B and C at least one more time.

E. Resolve the Violations and Transform the Data

For each assertion violation write python code to resolve the violation according to your entry in the “how to resolve” section above.

Output the validated/transformed data to new files. There is no need to keep the same, awkward, single file format for the data. Consider outputting three files containing information about (respectively) crashes, vehicles and participants.