DataEng S22: Data Validation Activity

High quality data is crucial for any data project. This week you'll gain experience with validating a real data set.

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: I worked on a Japanese translation project that had translation errors. A native speaker helped validate these.

Response 2: While importing data into a database, it was with the wrong encoding and special characters showed up as junk. I reimported the data with the correct encoding to take care of this.

Response 3: There was an error during a MIDI data project where some corrupted data needed to be manually labeled.

Response 4: In MIDI music composition notes went through a process called quantization where notes were put exactly on the beats they needed to be on, but this process overcorrects sometimes and needs to be manually adjusted.

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”

   Record type 1 should have a serial number, crash month/day/year
   Record type 2 should have a vehicle ID
   Record type 3 should have a participant ID

2. *limit* assertions. Example: “Every crash occurred during year 2019”

   Latitude should be between -90 and 90
   Longitude should be between -180 and 180

3. *intra-record* assertions. Example: “Every crash has a unique ID”

   Record type 3 should have a participant ID and vehicle ID.

4. Create 2+ *inter-record check* assertions. Example: “Every vehicle listed in the crash data was part of a known crash”

   Each crash ID has at least one vehicle ID and participant

5. Create 2+ *summary* assertions. Example: “There were thousands of crashes but not millions”

   There are not more vehicles than participants.
   There are more crashes during the winter.

6. Create 2+ *statistical distribution assertions*. Example: “crashes are evenly/uniformly distributed throughout the months of the year.”

Each month's crash amount will be within one standard deviation of the mean.

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:

- The data contained no violations for any of the constraints I attempted to measure.

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.

The first step would always be revising or questioning my original assumptions, since I'm not an expert on this kind of data. For most violations my response would likely be to discard the values if I felt my assumptions were not incorrect. Depending on the importance of the value the data may still be usable and I might add a default value to indicate it is unknown.

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?

The data is a flat file which was probably originally 3 different data tables. In order to properly analyze it I needed to figure out what was relevant data for each “record type”. It also contains some default values for unknown values, such as using 99 to indicate a status code was unknown. The data also contains no invalid information as far as I could ascertain.

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.

(Written to crashes.csv, vehicles.csv, participants.csv)