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 can't think of any data set I've worked with that included errors. The one that I have worked with mostly just didn't have the data we thought it would have like a username. We discovered that when the count of the queried data and the total set was way off. We had to reconsider our query plans and used a different field.

Response 2: I personally have not since the data that I was handling already went through some form of data validation before reaching me. But if issues in the data arrive I would try to manually fix them if possible or if is too much trouble to fix, I would scrap that piece of data entirely.

Response 3: I worked with some manufacturing data during an internship and had to validate data. Part of what I did was make some simple statistical simulations of manufacturing units over time and from that was able to find some outliers to validate. The larger thing that I did was find outliers in training data and then work with stakeholders to find the most accurate information. To reduce future errors, I developed dashboards and notifications to ensure managers were aware of the data they were responsible for.

Response 4: I haven't worked with set of data that included errors. But while working for some project in my undergrad we needed to fetch data set of user rating so I had to check whether the data fetched was right for further calculations and I had used excel for data validation for example made sure a particular column had numerical value ranging from 0-5 (ratings).

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 DMV Crash Serial Number
    b. Every crash occurred in some county
2. *limit* assertions. Example: "Every crash occurred during year 2019"
    a. Every county exists in Oregon state
    b. Every crash should either be in a school zone or not in a school zone
3. *intra-record* assertions. Example: "If a crash record has a latitude coordinate then it should also have a longitude coordinate"
    a. Every crash has a unique Crash ID
    b. Every vehicle has a unique Vehicle ID
4. Create 2+ *inter-record check* assertions. Example: "Every vehicle listed in the crash data was part of a known crash"
    a. Every crash involves at least 1 vehicle
    b. Every vehicle record must be associated with at least 1 crash
5. Create 2+ *summary* assertions. Example: "There were thousands of crashes but not millions"
    a. The most crashes happen in metropolitan counties
    b. Sports car were involved in the most crashes

6. Create 2+ *statistical distribution assertions*. Example: “crashes are evenly/uniformly distributed throughout the months of the year.”
    a. There are more crashes on the holidays
    b. There are more crashes during rush hour

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:

- Crashes are not evenly distributed throughout the months of the year because the standard deviation of crashes in each month is 8.8
    - This violation will require a revised assumption
-
-
-

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?

This dataset includes data for three separate tables. The distribution of crashes of the crash table is not even.

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.

The assertion violation I found was on standard deviation for evenly or unevenly distributed data. This cannot be resolved by changing the data with python code.

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.