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: lack of strict type checking, empty cells, data type formats ; consolidating data was the biggest issue

Response 2:

Response 3:

Response 4:

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” -> True
   i. Every crash has its record id (#serial no) -> True
   ii. Every crash has its vehicle ID -> True
   iii. Every crash has its participant ID -> True
   iv. Every crash has longitude and latitude value -> True
2. *limit* assertions. Example: “Every crash occurred during year 2019” -> True
   i. Every crash occurred on Highway no 26. ->True
   ii. Every crash participant’s age range between 0 -125. -> False (there was a null value and the min age value showing is 0 and max age value 9; looks some kind of code value)
3. *intra-record* assertions: fields within a record must satisfy some relationship
   Example: “If a crash record has a latitude coordinate then it should also have a longitude coordinate” -> True
   “Every crash has a unique ID” -> True
   i. Every crash id has a serial no. ->True
   ii. For Longitude coordinates, minutes and seconds fields are also populated if the degrees field is populated.-> True
4. Create 2+ *inter-record check* assertions.
   Example: “Every vehicle listed in the crash data was part of a known crash” -> True
   i. Each crash ID has at least one vehicle associated with it-> True
   ii. Each crash ID has at least one participant -> True
   iii. Participant ID may or may not be associated with a Vehicle -> True

5. Create 2+ *summary* assertions.
   Example: “There were thousands of crashes but not millions” ->True
   i. Two or more vehicles are involved in a crash on average -> False
   ii. The number of crashes tends to be higher in weekdays than weekend -> True

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

    i. The number of crashes tends to be higher in the beginning and end of the year than in the middle of the year -> False
    ii. Crashes are uniform throughout the day. -> False

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:

- Every crash participant's age range between 0 -125. -> False (there was a null value, and the min age value showing is 0 and max age value 9; looks some kind of code value)
-
-
-

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?

Learning from the provided data set are:

Many information can be extracted from the data like how many crashes happened in year 2019 in highway 26. Other information's like during what time of the day or what month of the year records highest accidents. Information on exact location of crashes can be derived from latitude and longitude provided in degrees, minutes, and seconds. Number of crashes were more in weekdays compared to weekends.

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.

```
CrashesDF.to_csv('CrashesData.csv')
VehiclesDF.to_csv('VehiclesData.csv')
ParticipantsDF.to_csv('Participants.csv')
```