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:
Me - I write automated tests on a REST API and data platform at work so the tests I write often encountered data issues. Resolving a particular issue depends on the context; I might have to alter the test if my assumptions about the data were incorrect, or there might be a problem when ingesting/transforming the data higher up in the pipeline so I have notify the engineers that work on that code, or we might have to discuss with the customer whether or not the data they inputted is correct.

Response 2:
Andy - at my job where we deal with developmental boards, to make sure we can boot linux or perform read/writes, sometimes we deal with a set of data that has invalid memory addresses and to debug that we implement data integrity checks and tests and validation testing to make sure the data/errors are valid

Response 3:
Mohamed - at my job where we deal with developmental boards, to make sure we can boot linux or perform read/writes, sometimes we deal with a set of data that has invalid memory addresses and to debug that we implement data integrity checks and tests and validation testing to make sure the data/errors are valid

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”
2. *limit* assertions. Example: “Every crash occurred during year 2019”
3. *intra-record* assertions. Example: “If a crash record has a latitude coordinate then it should also have a longitude coordinate”
4. Create 2+ *inter-record check* assertions. Example: “Every vehicle listed in the crash data was part of a known crash”
5. Create 2+ *summary* assertions. Example: “There were thousands of crashes but not millions”
6. Create 2+ *statistical distribution assertions*. Example: “crashes are evenly/uniformly distributed throughout the months of the year.”

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

- Existence
  - Every crash had a lighting condition
  - Every crash had a weather condition
- Limit
  - Hour code should be within 00-23
  - Day number should be within 1-31
- Intra-record
  - If a crash has a city section ID, it should also have a county

    - If a crash involved someone who died as a result of a driver with a BAC of .08 to .80, then the alcohol involved indicator should be true.
- Inter-record
    - The number of occupants in a vehicle is equal to the number of participant rows with that vehicle ID
    - A participant’s vehicle ID is represented in a vehicle row
- Summary
    - Every crash has a unique DMV serial ID
- Statistical distribution
    - Crashes involving intoxication generally have more injuries

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:

I didn’t run into any assertion violations because I only had enough time to write two assertions. It took me the entire first class to understand the data and come up with enough assertions to satisfy the first requirement and then I spent a good chunk of the second class understanding how to use pandas.

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.