DataEng: Data Validation Activity

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 for this week.

High quality data is crucial for any data project. This week you'll gain some experience and knowledge of analyzing data sets for quality.

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. First (part A) you develop assertions about your data as a way to make your assumptions explicit. Second (part B) you write code to evaluate the assertions and test the assumptions. This helps you to refine your existing assertions (part C) before starting the whole process over again by creating new assertions (part A again).

Submit: In-class Activity Submission Form

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 for this assignment, two or more assertions in each category are enough.

1. Create 2+ *existence* assertions. Example, "Every record has a date field".

Every participant record has a vehicle ID
Every record have a crash ID
Every vehicle record has a vehicle ID
Every participant record has a participant ID

2. Create 2+ *limit* assertions. The values of most numeric fields should fall within a valid range. Example: "the date field should be between 1/1/2019 and 12/31/2019 inclusive"

The crash month should be a value between 1 and 12 inclusive.

The crash day should be a value between 1 and 31 inclusive.
Crash hour must have code between 00 and 24 or 99.

3. Create 2+ *intra-record check* assertions.

Vehicle occupant count = safety equipment used quantity + safety equipment unused quantity + safety equipment use unknown quantity

Total persons involved count = total non-fatal injury count + total fatality count

Combination of month, day and year must represent a valid date

4. Create 2+ *inter-record check* assertions.

Total vehicle count = total number of vehicle records for this crash.
Total serious injury count = total number of participant records with injury severity value 2

5. Create 2+ *summary* assertions. Example: “every crash has a unique ID”

Every crash has a unique Crash ID
Every vehicle has a unique Vehicle ID

6. Create 2+ referential integrity insertions. Example “every crash participant has a Crash ID of a known crash”

Every crash participant has a Vehicle ID of a known Vehicle
Every crash vehicle has a Crash ID of a known crash

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

Crashes are evenly distributed throughout the day
Crashes are evenly distributed throughout the month
Crashes are normally distributed over the range of ages.

B. Validate the Assertions

1. Now study the data in an editor or browser. If you are anything like me you will be surprised with what you find. The Oregon DOT made a mess with their data!

2. Write python code to read in the test data and parse it into python data structures. You can 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. Again, pandas makes it easy to create and execute assertion validation code.
4. If you are like me you'll find that some of your assertions don't make sense once you actually understand the structure of the data. So go back and change your assertions if needed to make them sensible.
5. Run your code and note any assertion violations. List the violations here.

Every record has a unique crash ID, every record has age, and every has a crash year. These are examples of previous assertions I had that caused violations. I made those assertions with a false understanding of the table and I have made changes accordingly.

Lots of participant records do not have an known vehicle ID

The age data didn't follow the definition from the manual, and there are a large number of 2s and 9s.

Vehicle occupant count = safety equipment used quantity + safety equipment unused quantity + safety equipment use unknown quantity - these should all be attributes of the Vehicle Table but they aren't based on the data given.

Total vehicle count = total number of vehicle records for this crash.

Total serious injury count = total number of participant records with injury severity value 2

C. Evaluate the Violations

For any assertion violations found in part B, describe how you might resolve the violation. Options might include "revise assumptions/assertions", "discard the violating row(s)", "ignore", "add missing values", "interpolate", "use defaults", etc.

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

If you chose to "revise assumptions/assertions" for any of the violations, then briefly explain how you would revise your assertions based on what you learned.

I noticed that there are a lot of missing rows, and I learned that the crash data is actually three tables: crash, vehicle, and participants separated by record ID. I now have to revise my assertions to make sense within the specific table. For example, an assertion like every record should have a vehicle record doesn't make sense anymore.

Ignore rows violating the age assertion

Assign the total vehicle count to the correct value based on the number of vehicle records associated with its crash ID.

Revise - Every crash participant has a Vehicle ID of a known Vehicle - because actually not all participants should have a vehicle ID, some participants could be pedestrians.

Ignore rows violating the vehicle occupant count assertion

Ignore rows violating the total serious injury count assertion

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 ABCD iteration?

The dataset should be split into three tables: crash, vehicle, participant. It's a one-to-many relationship between crash and vehicle & participant and vehicle and participant.

Next, iterate through the process again by going back to Step A. Add more assertions in each of the categories before moving to steps B and C again. Go through the full loop twice before moving to step E.

E. Resolve the Violations

For each assertion violation found during the two loops of the process, write python code to resolve the assertions. This might include dropping rows, dropping columns, adding default values, modifying values or other operations depending on the nature of the violation.

Note that I realize that this data set is somewhat awkward and that it might be best to "resolve the violations" by restructuring the data into proper tables. However, for this week, I ask that you keep the data in its current overall structure. Later (next week) we will have a chance to separate vehicle data and participant data properly.

E. Retest

After modifying the dataset/stream to resolve the assertion violations you should have produced a new set of data. Run this data through your validation code (Step B) to make sure that it validates cleanly.

Submit: In-class Activity Submission Form