Data Validation Activity

A. Create Assertions

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

a. Fields like BAC Test Results Code, Alcohol Use Reported, Drug Use Reported, Participant Marijuana Use Reported, Participant Striker Flag are completely null. Every Crash ID has a Record Type.
b. Records with serial# only have Crash date, Crash month and Crash year, Weekday code, crash hour, country code, city section ID, urban area code, functional class code, NHS flag, Highway number.
c. Fields with record type = 3 only have participant ID field.

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”

a. Record Type field are in the range of 1 to 3.
b. The Crash year field should be 2019.

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

a. Total Non-fatal injury count = Total Suspected serious injury (A) count + Total Suspected Minor injury (B) count + Total possible injury (C) count – Total Fatality count
b. Total persons not using safety equipment = Total count of persons involved – Total persons using safety equipment

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

a. Mileage type is only for crashes that occur on the state highway system.

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

a. Crash ID has a unique serial number.
b. Participant ID is unique.

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

a. Every participant ID has a crash ID, vehicle ID,
b. Every participant ID has a participant seq for a crash.

7. Create 2+ *statistical distribution assertions*. Example: “crashes are evenly/uniformly distributed throughout the year.”
a. Here most of the crash types are 2 and 3.

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 make it easy to create and execute assertion validation code.
4. If you are like me, you will 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.

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.

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?

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