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".

   Assertion 1.a: All Records must have a record_type and the record_type should be either 1, 2 or 3

   Assertion 1.b: All record must have a non-null crash_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”

    Assertion 2.a: Data in Crash month field should fall within range 1 t0 12.

    Assertion 2.b: Data in Week Day Code field should fall within range 1 t0 7.

    Assertion 2.c: When entered, Latitude Degrees must be a whole number between 41 and 47, inclusive

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

    Assertion 3a: Total Count of Persons Involved = Total Pedestrian Count + Total Pedalcyclist Count + Total Unknown Count + Total Occupant Count.

    Assertion 3b: Total Un-Injured Persons Count = total number of persons involved - (the number of persons injured + the number of persons killed)

    Assertion: Date should not be future

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

    Assertion 4a: Total crash should not vary more than 50% month on month

    Assertion 4b: Latitude Minutes must be null when Latitude Degrees is null and Latitude Seconds must be null when Latitude Degrees is null

    Assertion 4c: Distance from Intersection must = 0 when Road Character = 1 and Distance from Intersection must be > 0 when Road Character > 1

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

    Assertion 5a: Check if all participant has unique id

    Assertion 5b: Combination of Serial number + County + Year is unique

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

    Assertion 6a: Each participant id has a crash id

Assertion 6b: Every crash has a known lat long location

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

   Assertion 7a: Crashes should be normally distributed across all months

   Assertion 7b: Crashes should be normally distributed throughout the day

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.

- *Month of crash, latitude degrees, minutes and seconds all have null values.*

- *Some rows are missing participant id when checking referential integrity with crash id.*

- *During Inter record check assertions, both the Total Un-Injured Persons Count and Total Count of Persons Involved are violated.*

- *During Summary Assertions, serial number, county and year are not available for all the rows.*

- *During Referential integrity assertion, not all rows have participant id and not all rows have lat long values.*

- *The month of occurrence for crashes are not normally distributed.*

- *The given crash hour values are not always within the 0 to 24 hr range.*

- *The Crash hour values are not normally distributed.*

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.

- *Ignored null values in crash month while doing limit assertion.*
- *Provided default values for Total Un-Injured Persons Count, Total Count of Persons Involved, the number of persons injured, the number of persons killed, Total Pedestrian Count, Total Pedalcyclist Count, Total Unknown Count and Total Occupant Count as 0.*
- *The intra record assertion failed rows are ignored for now for deleting those rows may remove too many rows.*
- *No defaults were given for serial number, year and county combination checks. Since this problem should go away when we isolate the crash data.*
- *Ignore records without participant id to assert each participant id has a crash id.*
- *Ignore records with record type 2 or 3, to assert each crash has a known latitude and longitude*
- *Provide a tolerance limit of 0.25 for distribution skewness for crash month distribution.*
- *Provide a tolerance limit of 0.25 for distribution skewness for crash hour distribution.*
- *Default missing crash hours to 0 and crash hours > 24 to 0.*

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?

- *Crash Data, Vehicle Data and Participant data are 3 different tables but given as a single table.*

- *Age column has only values from 1 to 9. It is definitely erroneous.*

- *Participant Type is one good column that helps in validating many other columns. But the column seems to be erroneous. (Expected to have 1 char values but all the values found are 2 chars). This limited me from testing many validations for other dependent columns.*

- *Columns involving various total counts do not tally well when math is applied across multiple columns.*

- *Data is given only for the year 2019. This constraints from doing right statistical inference.*

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