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:
No, I have not. I worked alone, so my response is the only one.

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”
    a) Every crash should have a county code
    b) Every crash should have a date
2. *limit* assertions. Example: “Every crash occurred during year 2019”
    a) Every crash occurred within a lat/long range that the highway exists within
    b) Crash week code should be between 0 and 7
3. *intra-record* assertions. Example: “Every crash has a unique ID”
    a) Every crash with severity of 2 should have at least 1 fatality??????
    b) Every type 3 record has a value in the “Age” column
4. Create 2+ *inter-record check* assertions. Example: “Every vehicle listed in the crash data was part of a known crash”
    a) Every type 1 record should have at least 1 type 2 or 3 record
    b) Total vehicle count should equal number of type 2 records for that crash ID
5. Create 2+ *summary* assertions. Example: “There were thousands of crashes but not millions”
    a) The number of crashes should not greatly exceed the number of vehicles
    b) The total number of crashes should not be more than 100,000
6. Create 2+ *statistical distribution assertions*. Example: “crashes are evenly/uniformly distributed throughout the months of the year.”
    a) Crashes are more common in the winter months than in the summer months
    b) Most crashes should cause death or injury

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:

- Error, fatal crashes should be rare
- Error, fatal crashes should have fatalities
- Error, should be as many type 2 records as value of 'Vehicle Count'
- Error, type 1 record should have at least one type 2 or 3 record

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.

- Error, fatal crashes should be rare:
  Turns out I was wrong about this, fatal/injury causing crashes are more common. This makes sense considering we're working with a highway.

- Error, fatal crashes should have fatalities:
  This error occurs sometimes but not others, which makes me think it's an issue with the data itself. As such I decided to just purge the offending data and move on.

- Error, should be as many type 2 records as value of 'Vehicle Count':
  I think this also might be an issue with the data, since the manual says the following about 'Vehicle Count': *The number of vehicles involved in this crash, excluding phantom or other non-contact vehicles. This derived field is calculated based on the number of vehicle records entered for this crash.* When I looked through the data manually, it did appear that there was something wrong here. Many type 1 records listed a vehicle count of 0 even when there were multiple type 2 records with that same crash ID. Since it was easy to calculate the number of type 2 records with the ID, I decided to just update the value.

- Error, type 1 record should have at least one type 2 or 3 record
  There was a problem with my code: I had an "or" instead of an "and", so if a type 1 record was missing either then there would be problems. Changing this to an 'and' fixed the error.