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:
In my previous database course, I encountered some data errors. At that time, I was managing the information of books in a library. When I ran some codes, I found that the format of some data was wrong, and the unit representation was inconsistent. Then I adjusted the format of these erroneous data and adjusted the units, and finally the code could run successfully.

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”
2. *limit* assertions. Example: “Every crash occurred during year 2019”
3. *intra-record* assertions. Example: “Every crash has a unique ID”
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.

1. Existence assertions: “Some crashes occurred in March”
2. Limit assertions: “Every crash occurred on highway 26”
3. Intra-record assertions: “Every crash requires at least one participant”
4. Inter-record assertions:
   “No more than 20 percent of crashes occur in June”
   “Each serial number is the part of the incident record as 1”
5. Summary assertions:
   “There are three record types in crash data”
   “In 2019, crashes took place for thousands of hours.”
6. Statistical assertions:
   “Crashes occur on average 20m away from intersections.”
   “The mode of the weekday on which the crash occurs is Wednesday.”

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:

- Revise assumptions/assertions
-
-
-

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.

Through this study of data validation, I first learned that there are seven kinds of assertions. I think step A is the hardest part. It requires us to create different assertions according to different situations, and the assertions are in line with the actual data. Step B I think is a relatively simple part, but by running the code to judge whether the

assertion we wrote is successful. If there is an error, proceed to step C to process our assertion. Repeating the three steps of ABC continuously can make our data more effective. I have not been exposed to pandas database before, and through this study, I have a deeper understanding of pandas database and learned a lot of code syntax.

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.

For statistical assertions, I assert “The mode of the weekday on which the crash occurs is Wednesday.” However, by running the code, I get a result not Wednesday, but Thursday, so I get one AssertionError. Then I modified the code to program Wednesday to Thursday and got a correct assertion.

```
print (df_1[‘Week Day Code’]. Median())                    //4
assert(print (df_1[‘Week Day Code’]. Median()==3))        //AssertionError

assert(print (df_1[‘Week Day Code’]. Median()==4))
```