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:

When we did the Book Management System, we need to scan the QR code for each book to get the ID. But, when we scan the QR code, we can only get two numbers in ID, however, we need to 10 numbers for each book. I forgot the specific code for this part, but we fixed them by using a detection system in order to get the right ID though the public information.

Response 2:

We also see some common mistakes. The virtual reality software parameters are incorrectly transmitted.

Response 3:

Error converting format when using JSON file, such as STR and list and dict format conversions. In particular, list of dictionaries in JSON.

Response 4:

None

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.

The new assertion:

1. *existence assertions. Example: "*Every crash has a serial code*"*
2. *limit* assertions. Example: "The maximum recorded type for each crash is 3"
3. *intra-record assertions. Example: "Each participant is a assigned a unique ID"*
4. *Create 2+ inter-record check assertions:*
    a. *Example: "Every crash requires three kinds of records."*
    b. *Example: "Every crash needs a participant"*
    c. *Example: "Every Serial# listed in the crash data was part of a known crash"*
    d. *Example: "The participant ID of each crash is required to retain the vehicle information of type 2"*

5. *Create 2+ summary assertions:*
    a. *Example: “Accidents usually involve about 1000 people a year.”*
    b. *Example: “At least 400 vehicles are involved in accidents each year”*
6. *Create 2+ statistical distribution assertions:*
    a. *Example: “The average age of the accident participants was around 30.”*
    b. *Example: “At least 90 percent of people have a valid driver's license when there's an accident ”*

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:
- Summary assertions(a):”Accidents usually involve about 1000 people a year”
- Statistical distribution assertions(a):”The average age of the accident participants was around 30”

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

- In Summary yassertions(a): The size of the assertion is problematic.
- In Statistical distribution assertions(a): When you calculate it, it's obviously not about 30. After looking at the data, the average age was between 0 and 8. Therefore, it is preliminarily believed that the data content is wrong.

- In Statistical distribution aassertions(b): At the time of the test, as many as 70 had no driver's license or were illegal.(Legal driver's licenses accounted for 28 percent of the total)

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?

- Although I already have a preliminary understanding of pandas, this exercise was done by using concrete examples, so I learned a lot about the specific use of the syntax, especially df.groupby(). This is the most difficult of all the syntax in my view.
- In addition to learning the syntax, in the assertion part, the real challenge is not to implement the concrete code but to provide the correct assertion based on the data. Therefore, I learned more about the relationship between data and data and how to express different assertions based on type before starting the 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.

The Modified code || Modify the assertion:
- Summary assertions(a):
  Modify the assertion:
  a) “Accidents typically involve at least 1,000 people a year”

- Statistical distribution assertions(a):
  Database error:
  a) The age distribution ranges from 0 to 9 rather than the true 0-99. Therefore, specific ages need to be modified and the general range should be between 12 and 60.
- Statistical distribution assertions(b):
  Not Sure:

a) This result surpasses my common sense. Considering that the legal proportion is too low, I think there are some errors in the data that need to be reviewed again.

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.

Generate 3 Files:

1) Crashes.csv
2) Vehicles.csv
3) Participants.csv

Python code:

‘’’

```python
df_crashes = df[df["Record Type"] == 1]
df_vehicles = df[df["Record Type"] == 2]
df_participants = df[df["Record Type"] == 3]

df_crashes.to_csv('crashes.csv')
df_vehicles.to_csv('vehicles.csv')
df_participants.to_csv('participants.csv')
```