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:

Sai: Most of the data I’ve worked with has been for class projects with data provided by a professor for that project. One example I can was last term in an NLP project translating Japanese patents to English. There were lots of errors such as translating things such as copywrite symbols.

Response 2:

Jared: I’ve had encoding differences such as UTF to ascii. So many characters show up as junk. So I redid the project using only UTF. Jared also had to deal with quantized music data to randomize it so it sounds more natural.

Response 3:

Christopher experienced errors in MIDI data for a machine learning project, and many tokens had to be manually labeled.

The data set for this week is [a listing of all Oregon automobile crashes on the Mt. Hood Hwy (Highway 26) during 2019](https://docs.google.com/spreadsheets/d/1GFBcW39v218FOjVZhsb5u-3LB7ok839BRuJA_2KwfT4). This data is provided by the [Oregon Department of Transportation](https://www.oregon.gov/odot) and is part of a [larger data set](https://tvc.odot.state.or.us/tvc/) that is often utilized for studies of roads, traffic and safety.

Here is the available documentation for this data: [description of columns](https://docs.google.com/spreadsheets/d/1AFa1p74257uTbb_cN8l2ztMPuFh2WGLq), [Oregon Crash Data Coding Manual](https://www.oregon.gov/ODOT/Data/documents/CDS_Code_Manual.pdf)

Data validation is usually an iterative three-step process.

1. Create assertions about the data
2. Write code to evaluate your assertions.
3. 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.

I found invalid data, for example. for Crash Id "1847540", if we count the unique vehicle ids the count is 4. but the count mentioned in "Total Vehicle Count" is 3

1. *existence* assertions. Example: “Every crash occurred on a date”

Every crash must have a Crash Day value.

1. *limit* assertions. Example: “Every crash occurred during year 2019”

Hours must be an integer between 0 and 24 or they should be 99 since 99 is the code for when the hour is unknown.

1. *intra-record* assertions. Example: “Every crash has a unique ID”

Every Crash ID in CrashesDF should be unique.

1. Create 2+ *inter-record check* assertions. Example: “Every vehicle listed in the crash data was part of a known crash”

Inter-Record Assertation 1: Every Crash ID should have a record in CrashesDF, VehiclesDF, and ParticapantsDF

Inter-Record Assertation 2: There should not be more vehicles than participants.

1. Create 2+ *summary* assertions. Example: “There were thousands of crashes but not millions”

Summary Assertation 1: There should not be millions of participants.

Summary Assertation 2: There should not be millions of crashes.

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

Statistical Distribution Assertion 1: There should not be radical anomalies in the crash hour distribution. Indeed, it looks like a normal distribution centered around 15:00.

Statistical Distribution Assertion 2: There should not be any radical anomolies for the months crashes occurred. As expected, there were more during winter, but everything looks as it should.

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:

* At first we thought that values of 99 violated the limit assertation that Crash Hour should be between 0 and 24, but then we realized that 99 is the code for hour unknown.
* No other violations were encountered.

For each assertion violation, describe how to resolve the violation. Options might include:

* **revise assumptions/assertions –** we did this after realizing that 99 was an intentional value with a specific meaning.
* 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.

I separated the tables before validation, have exported all as .csv files, and will upload all with this assignment.