FAA GMU Laser Study

AWS Glue DataBrew Recipe Files

# Background

AWS Glue Databrew was used heavily for the data cleaning/parsing task of the study. It came in handy to clean up a lot of natural language processing issues with messy data (type-o’s, incorrectly spelled words and words strung together), as well as helping to quickly change some of the terminology in the dataset to better group and standardize (e.g., “multi-color” was reported as “multi-color”, “multicolor”, “multi color”, “multi-colored”, “multicolored”, “multi colored”, “multicolorred”, “multiple colored”).

Many of the recipe steps in the DataBrew recipes were also unique to the dataset to pull apart entries that were merged together into long strings of text, or to just focus on a singular goal (parse out this piece of data) and may not work for other datasets in the future.

Many of these recipes were run in succession, using the output of previous recipes/jobs. This was due to a limitation within AWS Glue Databrew to handle only a certain amount of work. In most cases the maximum was ~ 110 - 125 recipe steps. For this reason, recipes that go together are numbered #a,b,c, etc. So basically, the process to run these will be:

* Upload initial file to S3
* Create a dataset in AWS Glue Databrew using the file uploaded to S3
* Create a project using the dataset and recipe (see below)
* Make any desired changes to the recipe via the project console, publish recipe
* Run Job
* Output of job will be saved as a csv file in S3
* Create dataset in DataBrew with new file, create project w/ recipe + dataset, etc.

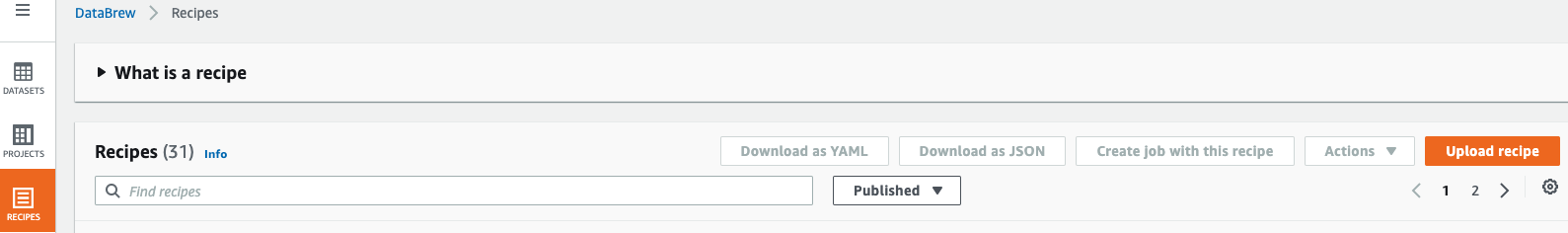
We also tried to split out the recipes by cleaning, then extraction; however, following extraction (parsing), some additional cleaning was sometimes needed on the extracted data, so you will see cleaning type functions within the extraction recipes.

# Recipes

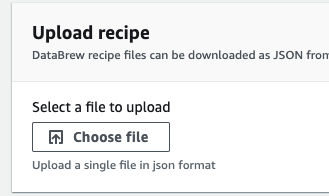
Recipes are all in JSON format and can be uploaded and run against a dataset in AWS. However, it’s important to note that the dataset must have the same fields/field names as the recipe references (e.g., you can’t run a function on a field that doesn’t exist).

To do this, you would first load the dataset into an S3 bucket.

Then go to AWS Glue DataBrew, select the “Recipe” icon on the left and then “Upload recipe” on the right”



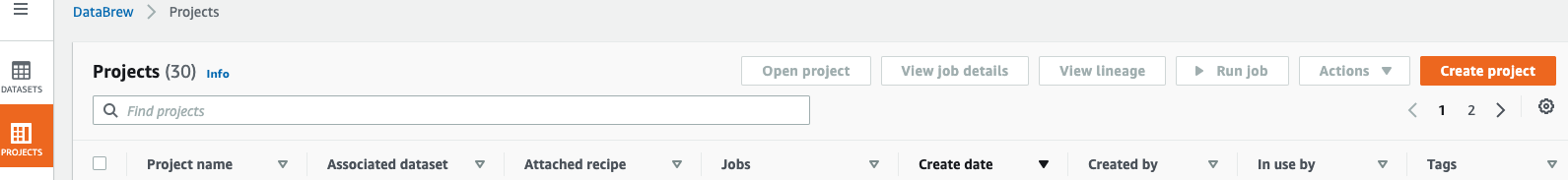
In the window that pops up, you would click on “Chose File” (you don’t have to enter Recipe Name and Description; the Recipe Name will auto-populate with the name of the recipe you upload)



Then “Create and publish recipe” at the bottom of the page



With the dataset and recipe uploaded, you can then create a project using those and either run a job with the recipe as is, or make changes to the recipe and then run the job against the data.



# Recipe Files and Descriptions

## Recipes and Number of Steps per recipe:



## Laser Color Parsing Recipes – 3 recipes, total of 199 recipe “steps”

2 recipes are used in succession to perform cleaning, followed by an extraction/parsing recipe.

1b and 1c could be combined into 1 recipe file to save time/cost. However, for simplicity, the cleaning and parsing files were kept separate. Also, when some of the more complex files (extraction, Regex) had too many steps, they tended to get overloaded and either ran too long or errored out.

* 1a – laser-color-part1 – 125 recipe steps
  + Recipe to perform data cleaning, formatting, and text replacement with a focus on the text remarks around the laser color
  + Input file was a file containing the Skywatch data X\_id, date, and remarks columns (after PII masking/removal)
  + Output of this Job is the input of 1b
* 1b-laser-color-part2 – 16 steps
  + Recipe to perform data cleaning, formatting, and text replacement
  + Uses output from 1a
  + Output of this job is the input of 1c
* 1c– laser-color-part3 – 58 recipe steps
  + Recipe to perform data extraction (parsing) from the remarks column
  + Uses the output of the 1b-laser-color-part2 job
  + Recipe first pulls out each color from the remarks field into a separate column, then merges the columns together, removes the word “laser”, and then clean up on the new column is performed to remove duplicate values.
  + To minimize the number of laser colors, any report that had multiple values (e.g., green and blue, red and green) was changed to multi-color
  + Output of this job is:
    - New column with laser color
    - Data input file for the Altitude parsing

## Aircraft Altitude Parsing Recipes - 14 recipes, total of 1594 recipe “steps”

This used the output of the Color parsing, so some cleaning was already done.

If these are run without doing that first, results may be different.

At a minimum, recommend changing the text to upper case and removing extra spaces, which was done in recipe 1a.

13 recipes are used in succession to perform formatting and cleaning, followed by an extraction/parsing recipe.

2m and 2n could be combined into 1 recipe file to save time/cost. However, for simplicity, the cleaning and parsing files were kept separate. Also, when some of the more complex files (extraction, Regex) had too many steps, they tended to get overloaded and either ran too long or errored out.

* 2a-laser-alt-part1 -136 steps
  + Uses the output of 1c-laser-color-part3 job
  + Output is used for 2b-laser-alt-part2
* 2b-laser-alt-part2 – 127 steps
  + Uses the output of 2a-laser-alt-part1 job
  + Output is used for 2c-laser-alt-part3
* 2c-laser-alt-part3 – 114 steps
  + Uses the output of 2b-laser-alt-part2 job
  + Output is used for 2d-laser-alt-part4
* 2d-laser-alt-part4 – 121 steps
  + Uses the output of 2c-laser-alt-part3 job
  + Output is used for 2e-laser-alt-part5
* 2e-laser-alt-part5 – 126 steps
  + Uses the output of 2d-laser-alt-part4 job
  + Output is used for 2f-laser-alt-part6
* 2f-laser-alt-part6 – 117 steps
  + Uses the output of 2e-laser-alt-part5 job
  + Output is used for 2g-laser-alt-part7
* 2g-laser-alt-part7 – 125 steps
  + Uses the output of 2f-laser-alt-part6 job
  + Output is used for 2h-laser-alt-part8
* 2h-laser-alt-part8 – 126 steps
  + Uses the output of 2g-laser-alt-part7 job
  + Output is used for 2i-laser-alt-part9
* 2i-laser-alt-part9 – 117 steps
  + Uses the output of 2h-laser-alt-part8 job
  + Output is used for 2j-laser-alt-part10
* 2j-laser-alt-part10 – 125 steps
  + Uses the output of 2i-laser-alt-part9 job
  + Output is used for 2k-laser-alt-part11
* 2k-laser-alt-part11 – 124 steps
  + Uses the output of 2j-laser-alt-part10 job
  + Output is used for 2l-laser-alt-part12
* 2l-laser-alt-part12 – 120 steps
  + Uses the output of 2k-laser-alt-part11 job
  + Output is used for 2m-laser-alt-part13
* 2m-laser-alt-part13 – 14 steps
  + Uses the output of 2l-laser-alt-part12 job
  + Output is used for 2m-laser-alt-part14
* 2n-laser-alt-part14-extract – 102 steps
  + Uses the output of 2l-laser-alt-part12 job
  + Output of this job is New columns with:
    - Extracted altitudes in feet
      * Altitudes reported as flight levels (FL) changed to feet
    - AGL/MSL where reported

## Laser Injury Parsing Recipes – 3 recipes, total of 201 recipe “steps”

2 recipes are used in succession to perform cleaning, followed by an extraction/parsing recipe.

(1b and 1c could be combined into 1 recipe file to save time/cost. However, for simplicity, the cleaning and parsing files were kept separate. Also, when some of the more complex files were (extraction) had too many steps, they tended to get overloaded and either ran too long or errored out)

* 3a – laser-injury-part1 – 87 recipe steps
  + Recipe to perform data cleaning, formatting, and text replacement with a focus on the text remarks around injuries reported
  + Input file was a file containing the Skywatch data X\_id, date, and remarks columns (after PII masking/removal)
  + Output of this Job is the input of 3b
* 3b-laser-injury-part2 – 75 recipe steps
  + Recipe to perform data cleaning, formatting and text replacement
  + Uses output from 3a-laser-injury-part1 job
  + Output of this job is the input of 3c-laser-injury-part3
* 3c– laser-injury-part3 – 39 recipe steps
  + Recipe to perform data extraction (parsing) from the remarks column
  + Uses the output of the 3b-laser-injury-part2 job
  + Recipe first pulls out different terms for the injuries reported (pain types, visual disturbance types, blindness types, etc. into separate columns, then merges the “like” columns together. Finally, all values in each column are replaced by a single number for the injury type (e.g., Pain = 4) and the remaining columns are merged into one column
  + Output of this job is New columns with injury type reported in terms of a number value(s) [Note: reported injuries are not mutually exclusive, some entries contained multiple injury types reported):
    - 0 = unknown
    - 1 = No injuries reported
    - 2 = Other
    - 3 = Visual Disturbance
    - 4 = Pain
    - 5 = Blindness

## Questions?

Contact: Char Burrage, [cburrage@gmu.edu](mailto:cburrage@gmu.edu)