**Aggregating algorithm**

As per the project brief, we have provided a python file (aggregate\_algorithm.py) which implements the algorithm laid out in “A generalized approach for producing, quantifying, and validating citizen science data from wildlife images” (Swanson et al.) in order to create aggregated classifications for the photos taken as part of the MammalWeb project.

Aggregate\_algorithm uses the pymysql library to handle the connection with the SQL database in which all the photo and classification data is stored.

The function provided should be run server side whenever a new classification is made for a photo by a ‘spotter’ on the mammalweb website.

**aggregate\_algorithm(photo\_id,connection,preBlank(=none),preflag(=none))**

connection is a pymysql connection object which gives access to a database. The algorithm finds all individual classifications for the photo with ‘photo\_id’ and combines them as per Swanson et al. into a row in a new database table, ‘aggregate’. As part of the implementation, we have to find which options represent the ‘blank’ options and which are ‘flags’, these values can be passed into the function if the algorithm is being run on a set of photos at once to save a small amount of calculation time. The parameters are ‘none’ by default and these values are calculated at runtime if not passed in.

**Outputs**

The evenness, (fraction) support and (fraction) blanks attributes of the output records measure the likelihood that the species given by the aggregate is correct. The age and gender are more simply the mode of the individual classifications.

The attribute ‘flag’ is a marker to show what state the aggregate for that photo has reached. These are laid out in Swanson et al. and have been added to the options table with their struc attribute as ‘flag’. Photos with a flag of ‘complete’ or ‘consensus’ should be removed from the pool of photos to be classified and the aggregate considered correct. Photos with a flag of ‘blank’ should also be removed as they almost definitely do not contain any animal. Photos with a flag of ‘incomplete’ should remain in the rotation as they require more classifications to be sure of the aggregates accuracy. Finally, if any flags are set to -1 an error has occurred and needs to be corrected. Flag values are contained in the options table with a struc attribute of ‘flag’.

Along with the aggregate\_classification function, the script includes a simple loop to run the algorithm on every photo in the database to completely populate the aggregate table.

Also included is test code to compare aggregate classifications with a gold standard data set.

This implementation uses the pymysql library available at <https://github.com/PyMySQL/PyMySQL> under a free use and distribution license.

Results

When the flag and blank options are precomputed and passed as preFlag and preBlank, the implementation runs in 0.12 seconds on an intel i5-4310M 2.7GHz. Without the precomputed values the average is 0.13. Around 83 minutes for current ~42k image database.

In total, 96% of the aggregate species are the same as the gold standard data.

When considering only the complete and consensus flagged aggregates, agreement is just over 99%.

The species and age fields are simply the mode of all the classifications but both agree with the gold standard on around 75% of photos. This is likely because