Knowledge Base Final Wrap Up

[**Performance Updates**](#_8qbyqxf6nmh5) **2**

[**Validation Updates**](#_rarg4fdbqc3o) **3**

[**How-to Technical Updates**](#_heqr91dheswk) **4**

[**Reference Links**](#_cr5lhfklzyfj) **5**

## 

## Performance Updates

**Caching**:  
Caching has been implemented - on the development server, uncached queries take up to 7 seconds, but queries that have been cached take ~0.4 seconds. To make sure that every value is cached, you can run the validation program below in the “Validation” section, which will run through every possible combination of gender, age, and ethnicity and cache distributions for them. To reset the caches, you can rerun the validation program with “override\_cache = 1”. This takes a few days to run! To reset the caches of a limited set of variables, you can instead pass a list to the “variables” value, which will then only run the validation on those variables. Note that caching is done whenever a new set of variables is encountered, and override\_cache is a parameter passed to the KB - if you need to make sure that a request is cached when you have reason to suspect that it might not be, you can run the request (with correct demographics, but otherwise random data as necessary) beforehand to make sure that it is cached.

You should not need to recache anything unless you add new data or alter the method by which distributions or the density plots are calculated. This is unlikely to occur for the planned remaining lifespan of the kb. Changing the cutpoints file will not necessitate resetting any caches.

**Multithreading**:  
The server has been reinitialized with multiple threads. Currently, we’re using 4 workers, one worker per CPU, with two threaded processes each. This can scale up as needed using more threads, or increasing the size of the AWS instance.

## Validation Updates

**Validation**:  
The program validate\_kb.py will generate a file that can be used for manual validation of KB variables. Currently, it generates a csv with the column headers ['Variable', 'Ethnicity', 'Gender', 'Age', 'Target\_Value', 'Interpretation', 'Percentile', 'Distribution\_Size']. Note that some ethnic populations do not have sufficient data to return a distribution, which you can see from the validation file - when querying the KB with one of these ethnicities, it will return an empty list for the distribution. The validation program can be run from anywhere, but you will need to modify the url and give it an appropriate filepath on the EC2 instance running the script. Additionally, when run on the KB server and naming the output “validation.csv” in the default path, the validation.csv will be viewable at /validation.  
  
**Viewing Cutpoints**:  
The /cutpoints url contains a basic table view with filtering to view the cutpoints for variables, for easy reference. Each row in the table is a set of demographics - ie age, gender, ethnicity, - a range of values, and the interpretation for that range of values for the set of demographic variables.

**Viewing Distributions**:

The /validation url is the same, but for viewing the validation data. Each row is a combination of variable name, ethnicity, gender, race, a sample value, an interpretation, percentile of the sample value, and the size of the distribution for the demographic set.  
  
**Validating Calculated Variables**:  
The calculated variables are not currently being validated like the other variables - because they have a much larger possible combination of inputs, they need a different method of spot checking. As a start, derived\_vars\_ranges.py contains a dictionary with the expected ranges/values for each input value.

## How-to Technical Updates

**How to change the API:**  
The API as it stands is detailed at the end of this [document](https://docs.google.com/document/d/1EzeDIWRDnbKNb_h_sxjA1bmemjS745MnbjPXBlNWg6Y/edit?usp=sharing). The functions “get\_json” and “process\_query\_string” in flask\_index.py will need to be modified so that they give the same variables to the “Normalizer” function as they currently are (and obviously needs to be done at the same time as the api is modified in Lab100).

## Reference Links

[Description of each file and API](https://docs.google.com/document/d/1EzeDIWRDnbKNb_h_sxjA1bmemjS745MnbjPXBlNWg6Y/edit?usp=sharing) - Goes through the list of files in kb and describes how they work. Additionally describes the structure of the API as of 5/13/2019  
[Summary of metrics](https://docs.google.com/document/d/1ZndIecIRAZSI_Z-jkgZbMoPK6j8oRNU0SrhWscWSRzA/edit?usp=sharing) - describes each metric used by Lab100, including what device is used and where population and interpretation data is sourced.  
[Bitbucket Link](https://bitbucket.org/MaxTomlinson/lab100kb/) - link to the bitbucket repository for the code. Requires permission - if you want to view this, you’ll need to make a bitbucket account and ask for view permission from Max Tomlinson (max.tomlinson@mssm.edu)

[Lab100kb Normalization Explanation](https://docs.google.com/document/d/1R-YbjQ9vpBTzKHhUzlkf-CWohusP8yCCurKS441mPug/edit?usp=sharing) - explains in detail how population data from NHANES is normalized for a patients demographics.