Batch Tester Framework Protocol

Author: Vyas Gupta (vyas100gupta@gmail.com)

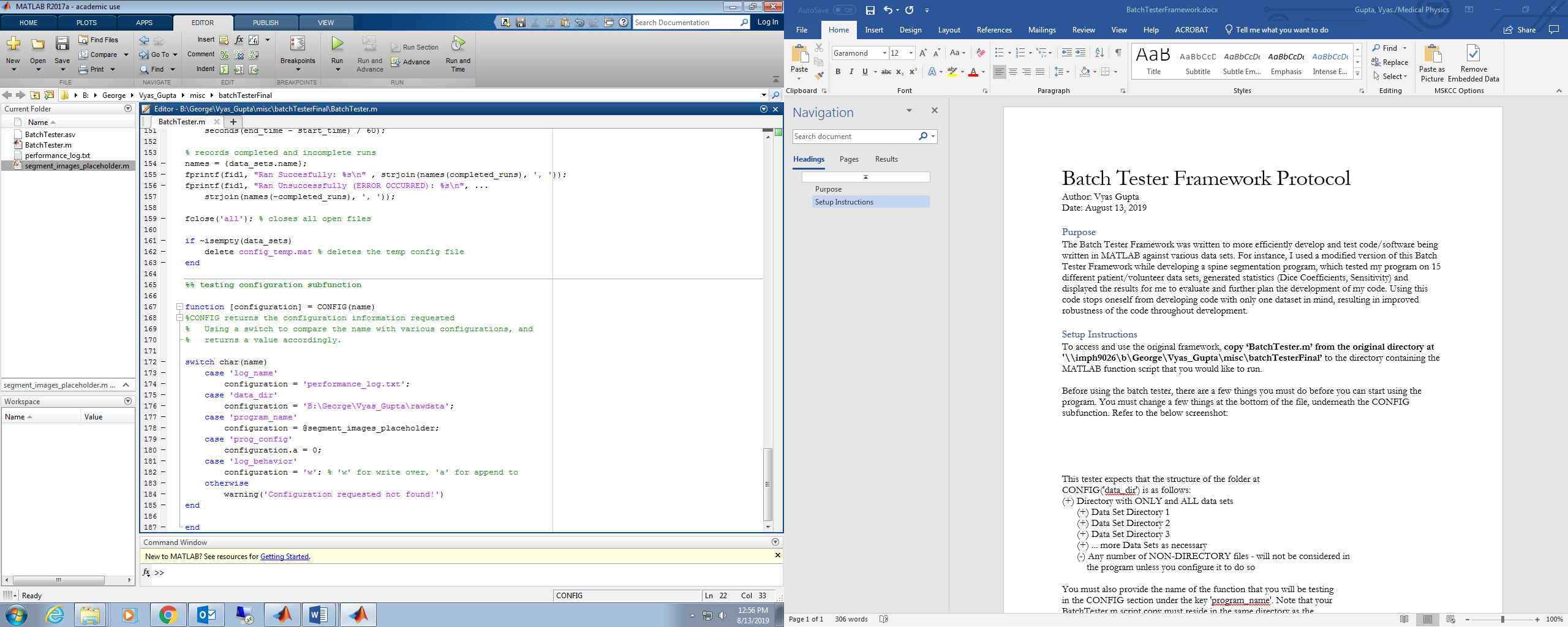
Date: August 13, 2019

## Purpose

The Batch Tester Framework was written to more efficiently develop and test code/software being written in MATLAB against various data sets. For instance, I used a modified version of this Batch Tester Framework while developing a spine segmentation program, which tested my program on 15 different patient/volunteer data sets, generated statistics (Dice Coefficients, Sensitivity) and displayed the results for me to evaluate and further plan the development of my code. Using this code stops oneself from developing code with only one dataset in mind, resulting in improved robustness of the code throughout development.

## Setup Instructions

To access and use the original framework, **copy ‘BatchTester.m’ from the original directory at '\\imph9026\b\George\Vyas\_Gupta\misc\batchTesterFinal’** to the directory containing the MATLAB function script that you would like to run. Before using the batch tester, there are a few things you must do before you can start using the program. You must change a few things at the bottom of the file, underneath the CONFIG subfunction. Refer to the below screenshot:



If you scroll to the bottom of the BatchTester.m script, this CONFIG subfunction should show up. In this subfunction, there are a couple changes you must make before being able to run your scripts.

Firstly, we must provide the tester with the proper function to test, found in the screenshot at line 178. In my case, you can see that in the upper left window, I have the ‘segment\_images\_placeholder.m’ script highlighted. This is a function that I would like to test, so underneath the ‘program\_name’ case, I give a pointer to the function by using the @ symbol, followed by the name of the function. This function should take a path and a configuration structure as its inputs, and output one structure, however this can be configured and changed easily (refer to ‘Further Configurations’ section).

Secondly, we must give a directory containing all the datasets that you would like to test, found in this screenshot at line 176. Copy the full path and paste it as a char array/string underneath the ‘data\_dir’ case. **Please note!** This tester expects that the structure of this folder is as follows:

(+) Directory with ONLY and ALL data sets (refer to ‘B:\George\Vyas\_Gupta\rawdata’)

(+) Data Set Directory 1

(+/-) Can contain anything, up to function to determine how to parse contents…

(+) Data Set Directory 2

(+/-) Can contain anything, up to function to determine how to parse contents…

(+) Data Set Directory 3

(+/-) Can contain anything, up to function to determine how to parse contents…

(+) ... more Data Set Directories as provided

(-) Any number of NON-DIRECTORY files - will not be considered in

the program unless you configure it to do so

After going through the above steps, you should be ready to test! If you would like to include more configurations, feel free to keep reading.

## Further Configurations

This code also provides a lot of room to be modified. For instance, I modified the original version to test my spine segmentation code and calculate statistics comparing my codes mask and a provided manual ground truth mask. You can refer to this tester at '\\imph9026\b\George\Vyas\_Gupta\Shell-Cavity\SpineSegmentation\TestSegmentationBatch.m’.

Here are a few additional tips and ideas:

1. If you don’t want to change your function outputs and inputs, you can modify the Batch Tester code to accommodate for this. At the function call, found at line 126 in the original copy, you can change the inputs and outputs to match your function and then add some post processing if you would like.
2. When logging the performance of your code, some like to append new logging information to a long file, which others would rather overwrite/create a new file each time. Underneath ‘log\_behavior’ in CONFIG, one can change from ‘a’ (append) to ‘w’ (overwrite) as they prefer. Refer to line 182 in the above screenshot.
3. One can add configurations to the config structure that the function to be tested accepts. Underneath ‘prog\_config’ in CONFIG, you can add various values to the config structure, which the function to be tested can parse through and use as needed. Refer to line 180 in the above screenshot.
4. If you would like to have more values recorded from the program, store these values in output as a structure and then add fprintf statements following the function call at 126 to print these files to the performance log.