Using Shiny for fMRI Quality Assurance

An Wang and Natalie Koh

June 8, 2017

1 Overview

Performing quality assurance (QA) checks for pre-processed MRI data is an important and necessary step in standard MRI processing pipelines. Previously, QA at IBIC involved having someone (or more than one person) look over select parameters, images and movies rendered on a HTML page that was generated for each subject and each fMRI modality. Logging and flagging subjects with data that did not look right was recorded separately, sometimes using an excel spreadsheet or a Google Document.

To circumvent the tediousness of having to open up several HTML pages for many subjects and many different fMRI scans, and then creating and tracking a log sheet elsewhere, we have built a Shiny application to gather everything needed for fMRI QA into one place. Shiny is an open source R package that allows us to build web application frameworks with R. The Shiny QA application we have created is a graphical user interface that is easy to use and allows users to seamlessly switch between subjects and fMRI tasks and log their comments on a CSV file that can be looked at in the application itself.

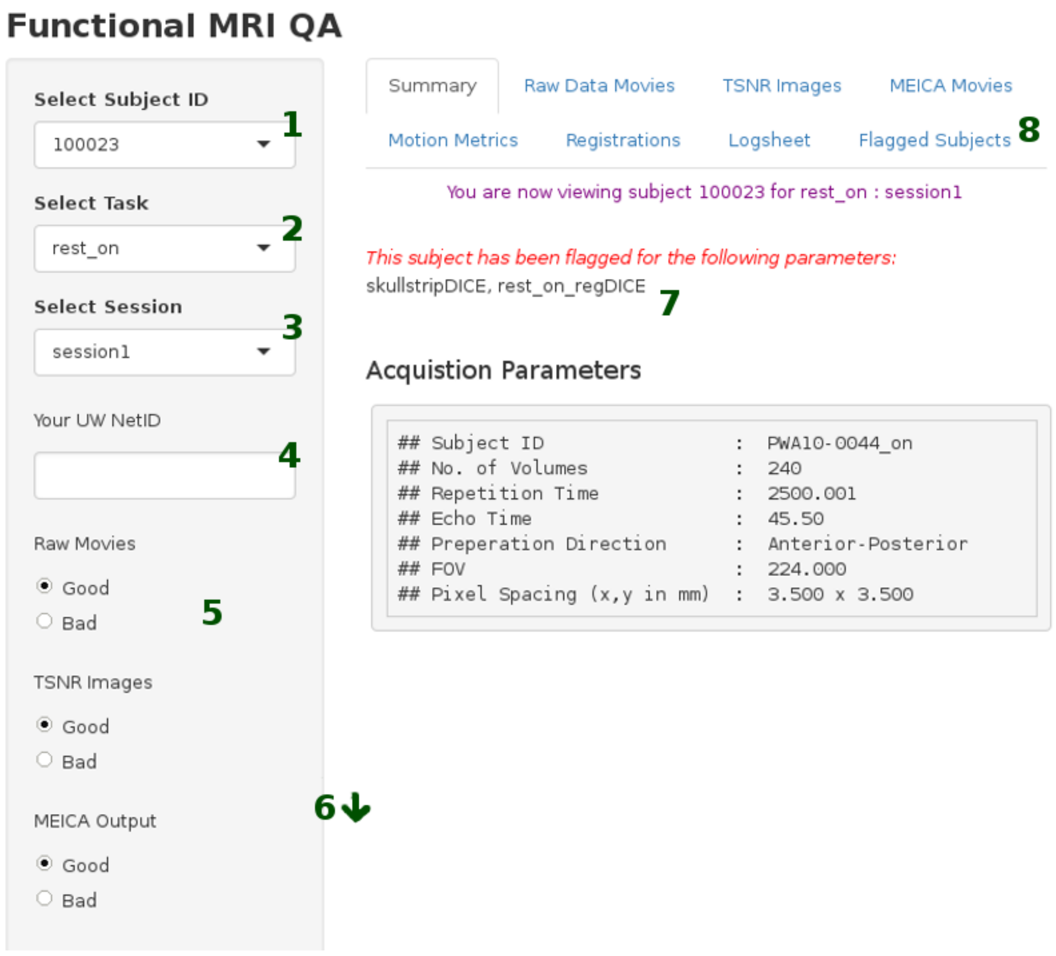
In this guide, we will cover how the application works, how to distinguish between "good" and "bad" images, and how to add warning checks to flag subjects with data that does not look right.

2 The Application

To launch the application, type the following into the terminal:

```
Rscript -e 'library(methods);
shiny::runApp("/mnt/panuc/IBIC_Pipelines/ShinyQA", launch.browser=TRUE)'
```

Below is an image that shows how the application looks like when it is first opened, and how it works:



Select Subject ID: 100023

Select Task: rest_on

Select Session: session1

Your UW NetID:

Raw Movies: [✓ Good] Bad

TSNR Images: [✓ Good] Bad

MEICA Output: [✓ Good] Bad

You are now viewing subject 100023 for rest_on : session1

This subject has been flagged for the following parameters:
skullstripDICE, rest_on_regDICE

Acquistion Parameters

```
## Subject ID                  :  PWA10-0044_on
## No. of Volumes              :  240
## Repetition Time             :  2500.001
## Echo Time                   :  45.50
## Preperation Direction       :  Anterior-Posterior
## FOV                         :  224.000
## Pixel Spacing (x,y in mm)   :  3.500 x 3.500
```

1. **Subject ID**: This is a drop-down menu that lists the subjects available for QA.

2. **Tasks**: "Tasks" may include resting-state fMRI or breathhold fMRI, or some other task fMRI.

3. **Session**: Currently, all subjects have only session1; if the study in

longitudinal, more sessions will be added as the study progresses.

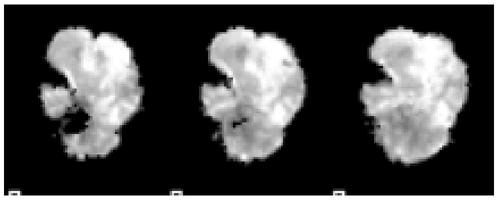
4. **UW NetID**: Input your NetID here. This is necessary so we can keep track of the individuals who are looking at the QA reports. This will be submitted along with your comments to the QA logsheet.
5. **Good/Fair/Bad**: For each measure, find the corresponding tab and select "good", "bad", or "fair" based on quality of the data.
6. **Submit a log**: Scroll down to the bottom of the sidebar, and click SUBMIT. If you accidentally submit an incorrect log, go to ShinyQA/output/ and remove your most recent entry.
7. **Warnings**: If warnings show up, pay especially close attention to the parameters or data that are flagged. These images are more likely to be problematic.
8. **Flagged Subjects**: This tab lists all subjects that have data that has been flagged.

Next, we will take a look at how to distinguish between "good" and "bad" images.

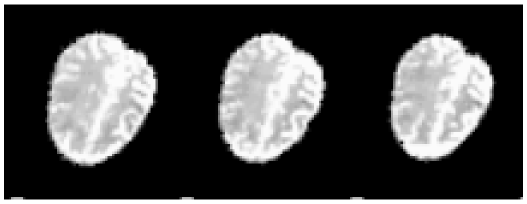
3 Checking the Quality of Data

In the section, we show some examples of bad and good images. The bad images may need to be re-preprocessed.

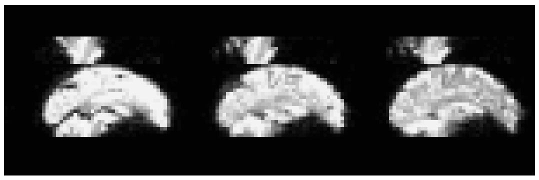
Raw Data Movies:



NOT an issue: Significant chunks of the brain are omitted in some of the movies. This is because of the way the brain is sliced, and does not need to be fixed.

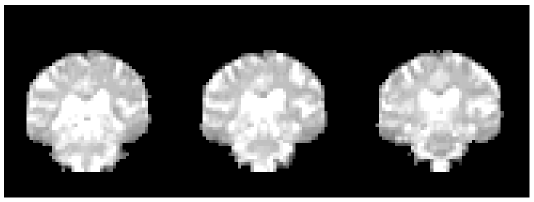


Issue: The brain is significantly tilted.
Fix: Make sure that the brain has been reoriented to standard space.



Issue: A movie show wraparound effects as it runs.
Fix: See documentation on how to fix multiecho reconstruction failure erros on UDALL dropbox.

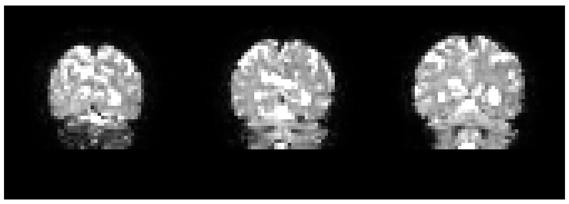
This is a usable image. No artifacts are present, and the whole brain is represented.



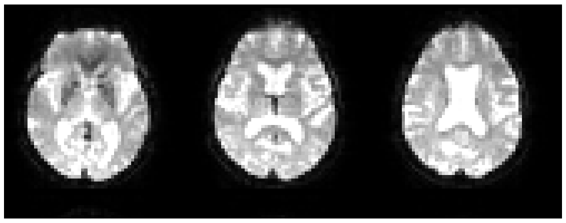
TSNR:

Bad TSNR images may appear blurred. Contrast may be incorrectly adjusted so that distinct brain structures are indistinguishable.

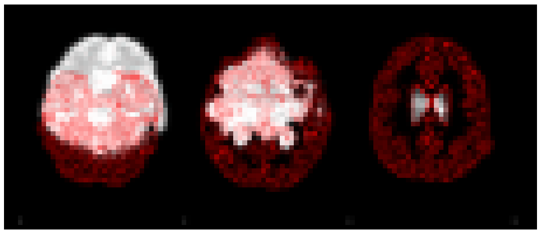
Brain structures are clear, and there is no obvious blurring. This is a good image.



This is a good image. Contrast here is good, with clarity in regards to brain structures.

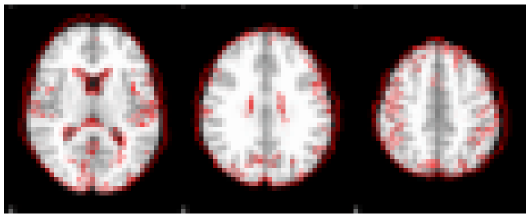


Registration:

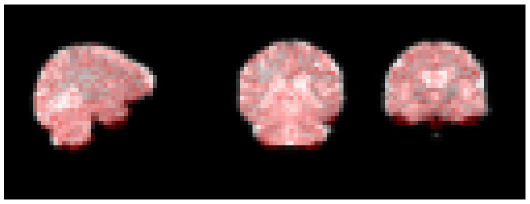


<u>**Issue:**</u> Registration appears to barely try matching the actual brain.
<u>**Fix:**</u> Re-run the registration processing. If the tracing is still terribly off, PROC

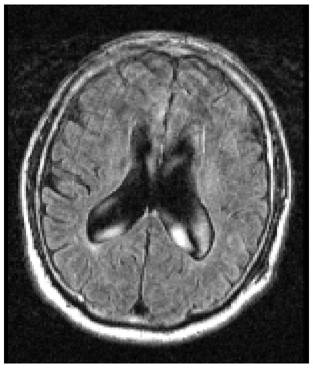
Registration looks *about* right. This image is acceptable.



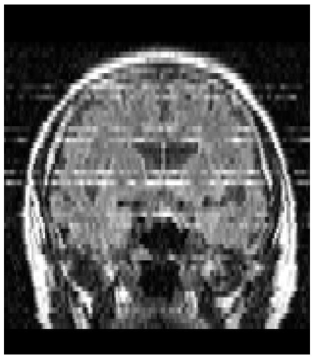
Registration follows the brain's contours fairly accurately. This image is good.



Other issues to look out for:



T1 images: Some ringing and blurriness can be perceived in the brain. This is due to motion during the T1 scan. No fix is currently available.



T1 images: Noise from the scanner itself appear on the image. Nothing can be done about these static-y artifacts, so this image is unusable.

For even more examples of what can go wrong with brain images, see this.

4 Warnings

Skullstrip & Registration DICE:
e.g. skullstripDICE, rest_on_regDICE
For Skullstrip DICE, the warning indicates that the process of separating skull from brain in T1 images might not have been entirely successful. For registration DICEs, the warning indicates that the registration of the fMRI images to MNI space or study-specific template spaces are not very good and may need

to be redone via some other method.

Motion Metrics:
e.g. axcpt_on_dvarsvals_e002_mean, abs_mean_displacement, percent_outliers
Warnings generated for motion metrics are primarily an indication that something may be wrong with task-relevant brain images due to too much motion; in movies, look for excessive motion. In still images, look for blurriness.

Scan Parameters:
e.g. ME_RS_off_slices, ME_Task_on_echoes
These warnings indicate that the scan acquisition parameters for this subject's scan were incorrectly set or different from what they should be.

5 Updating/Adding Warnings to the App

Each subject's warnings appear near the top of their summary page. Additionally, warnings for all subjects can be observed in the flagged subjects tab. These warnings are read from the file `QA/SUBJECTID_QA_stats.csv` from each subject's directory. The file is structured as follows:

```
measure,data,flag
RS_off_slices,37,0
RS_off_echoes,3,0
RS_off_dyn_scans,240,0
RS_off_FOV_RL,224.000,0
...
```

and so on. When a flag = 1 (see third column), the corresponding measure's name is displayed in the app as a warning. The data column contains the measure's quantitative value.

To generate these .csv files, run `make QA/SUBJECTID_QA_stats.csv` from the terminal from each subject's directory. Alternatively, one can also call the `generate_QA_stats` script directly with two arguments `SUBJECTID` and `SESSIONNO` where the latter is the desired session number. Here is the default location of the QA csv-generating script:

```
/mnt/panuc/IBIC_Pipelines/ShinyQA/generate_QA_stats
```

Altering flag thresholds: All thresholds are defined at the beginning of the script. To change the thresholds, simply edit these existing fields and regenerate all subjects' csv files after the update.

Adding potential warnings: Follow the output format (measure,data,flag) as shown above when adding to the script `generate_QA_stats`. This script generates flags in two stages, for task-independent and then task-dependent measures. Measures are task-dependent if they can be categorized as either `rest_on`, `rest_off`, `axcpt_on`, or `axcpt_off`. Add flag computations under the appropriate stage. Then re-generate the csvs for all subjects under the new script.

Future Improvements: Whenever a QA stats csv file is updated, it is regenerated from scratch. As the flags being generated are currently inexpensive to compute, regeneration is quick. However, if more time-consuming flags are to be computed in the future, it would be a good idea to look into caching such flags in such a way that they need not be recomputed.