



VisualQC

1.0 PURPOSE

This document is designed to give an overview of how to install and utilize VisualQC to rate FreeSurfer recon-all T1 cortical segmentation output, as well as common segmentation errors to look for. The resulting ratings can then be used to either exclude erroneous segmentations from further or analysis, or mark which ones should be edited, following the FreeSurfer recon-all troubleshooting guide:

<https://surfer.nmr.mgh.harvard.edu/fswiki/FsTutorial/TroubleshootingData>

2.0 DEFINITIONS

ENIGMA QC: Protocol developed by the University of Southern California for identifying and rating common errors that occur in FreeSurfer cortical segmentation.

FreeSurfer: Popular open-source, free software that provides cortical segmentation and parcellation of T1-weighted structural images.

VisualQC: Visualization and rating software for FreeSurfer cortical segmentation output.

3.0 PROCEDURES

3.1 Installation

3.1.1 Prerequisites: VisualQC requires Python 3+ (available at: <https://www.python.org/downloads/>) and FreeSurfer (available at: <https://surfer.nmr.mgh.harvard.edu/fswiki/DownloadAndInstall>) to run. It is highly recommended that you run VisualQC in a Linux environment; VisualQC has not been tested and is not supported in a Windows environment. If you choose to try running VisualQC in Windows, you will not be able to create surface visualizations as FreeSurfer will not run in a Windows environment and you will need to add VisualQC to your PATH variable, after installation, in order to run. To do this, either open a cmd window (search 'cmd' under programs) and run this instruction:

```
setx path "%path%;c:\pathtoVisualQC"
```

where the text `pathToVisualQC` needs to be replaced by actual path to VisualQC on your system **or**, by opening Control Panel > System > Advanced system settings > Environment Variables... Click Edit beside the system variable marked Path to modify the path. At the end type a semicolon to indicate a new path before copy/pasting the full path to VisualQC's files.

3.1.2 VisualQC can be found on GitHub at the following link:

<https://github.com/raamana/visualqc>.

To install: run this command in your terminal:

VisualQC User Manual

```
pip install -U visualqc
```

If you do not currently have pip (automatically included with python version 3.4 and greater) installed on your machine, you can follow the installation instructions on this site: <https://pip.pypa.io/en/stable/installing/>

Depending on your version of pip, you may receive this error:

```
from pip import main
ImportError: cannot import name main
```

If this occurs, include `python -m` at the beginning of the previous code line and then it should install without issue.

- 3.1.3 To ensure that VisualQC was installed correctly, run the following command in your terminal:

```
visualqc_freesurfer -h or use the short form: vqcfs -h
```

This should bring up VisualQC's help menu, including descriptions for all possible flags. If this does not happen and you receive an error, check that:

- 1) VisualQC has been correctly installed. If you run `pip install -U visualqc` again, you should get a message including: Requirement already up-to-date: visualqc
- 2) If running on Windows, check to make sure that visualqc is in your PATH variable (type path into cmd or check the Path variable under Control Panel > System > Advanced system settings > Environment Variables...)
- 3) Check that you have python 3 installed on your computer and that it is in your PATH. If you have both python 2 and 3 installed (you may get a message like this: `NotImplementedError: visualqc_freesurfer requires Python 3 or higher!`), you may need to set up a virtualenv to ensure that VisualQC is running on python 3. For instructions on how to set up and run a virtualenv, use <https://docs.python.org/3/tutorial/venv.html>. Once you have a virtualenv running with python 3, you will have to re-install VisualQC.

3.2 Running VisualQC

- 3.2.1 VisualQC can be opened using this command:

```
vqcfs -f </path/to/freesurfer/output/>
```

This will open VisualQC for all viable subjects within the given path.

- 3.2.2 To view only a sub-group of the subjects within the given folder, create a text document containing the names of all of the subjects you want to review, with one subject per line. Make sure to include the full subject name, as given on their FreeSurfer output folder:

E.g.

Subj_001

Subj_002

Subj_003

⋮

And submit to VisualQC using the `-i` flag.

```
vqcfs -f </path/to/freesurfer/output/> -i <list_of_subject_names.txt>
```

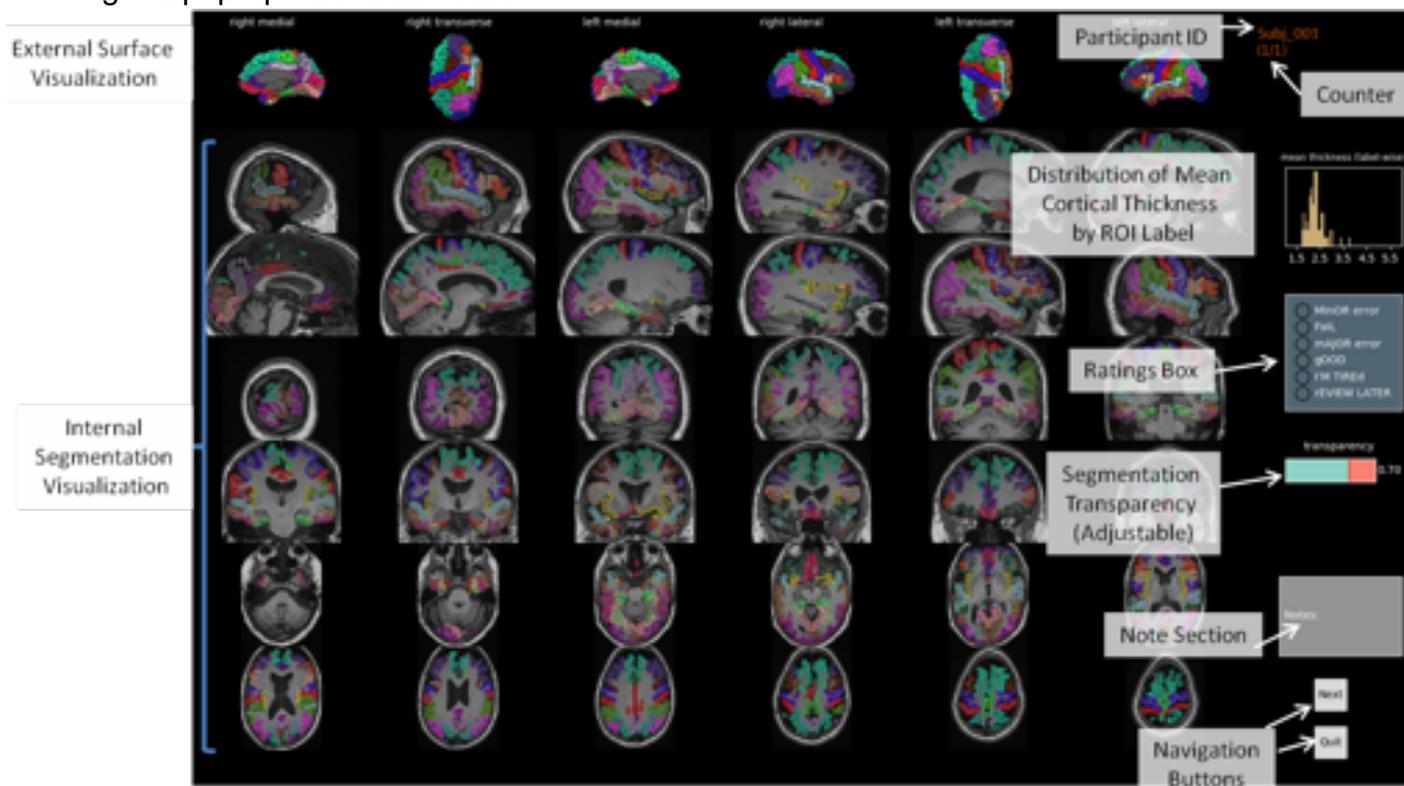
Note: you may need to disable outlier detection using the `-old` flag in order for VisualQC run, especially if only looking at a few participants. (Outlier detection is currently being tested and will be available in later releases)

3.2.3 Calling `vqcfs -h` will give you a complete list of features you can adjust when visualizing or by going to the following link:

https://raamana.github.io/visualqc/cli_freesurfer.html

3.3 The Interface

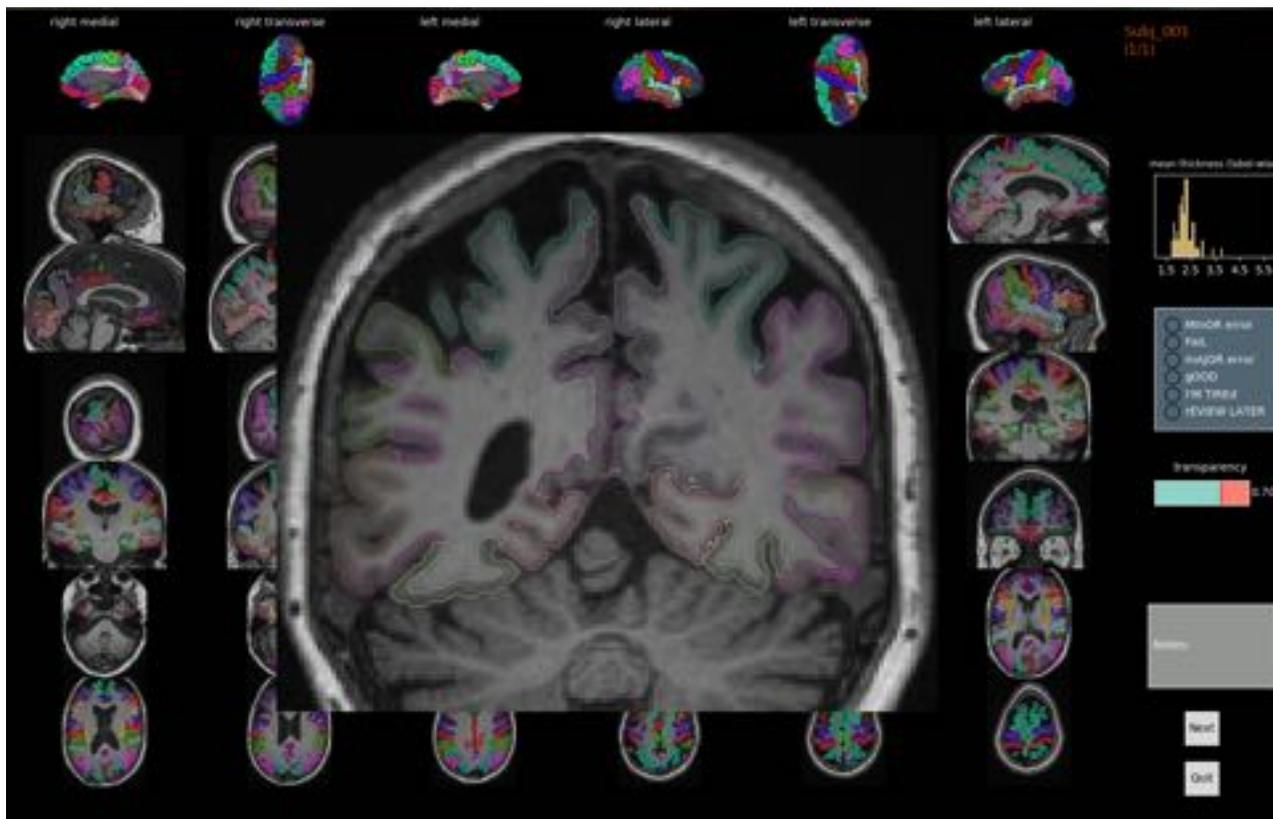
3.3.1 After VisualQC has finished producing the necessary visualizations, you should get a pop-up screen like this:



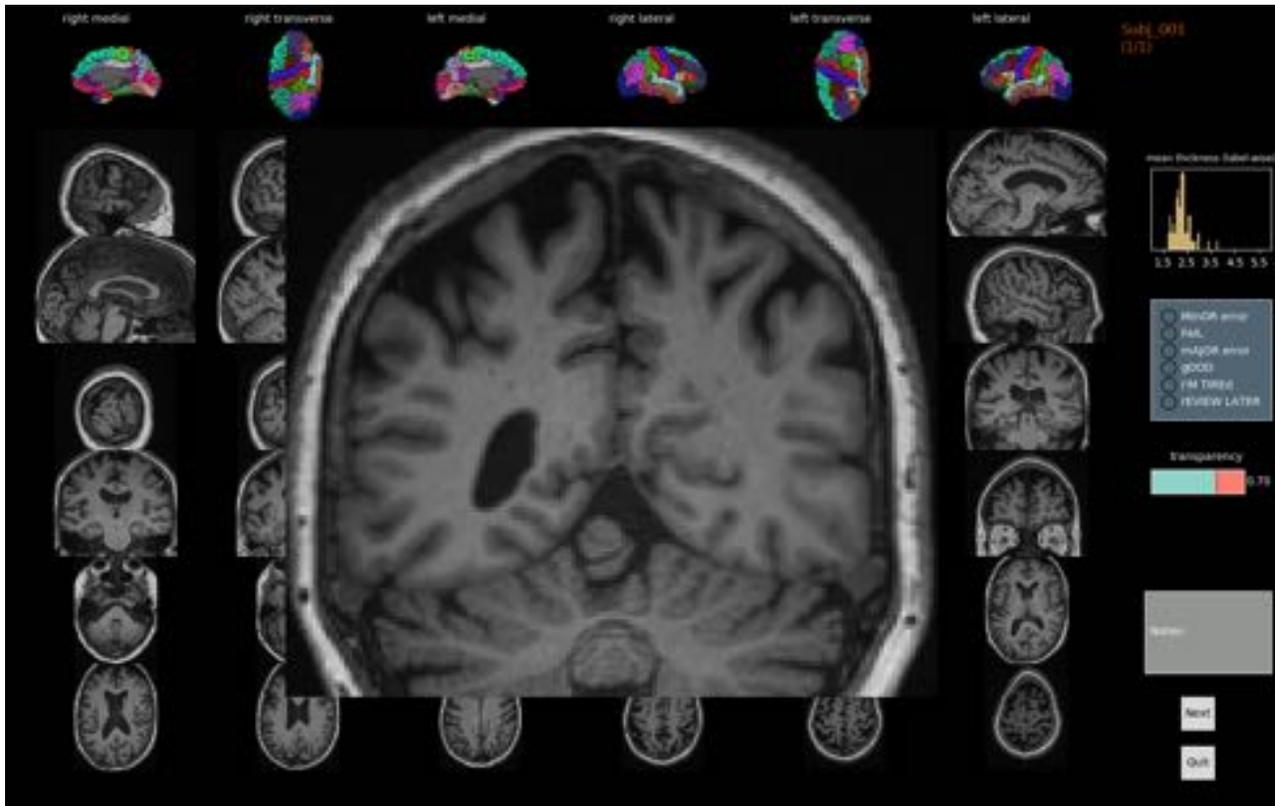
Note: make sure to adjust your screen brightness, contrast and angle so that you have the best outline-to-scan contrast before beginning. Some of the outlines may be similar in hue and tone, to the underlying mri scan and can be difficult to discern, especially under poor lighting conditions.

VisualQC User Manual

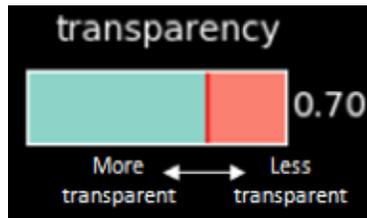
3.3.2 To zoom in on a slice, double click on a thumbnail and an enlarged view of the selected slice will appear:



3.3.3 To toggle the segmentation on and off, you can press 't' on the keyboard to get the below view; this can help in borderline cases where the exact structure of the pial surface can be difficult to see with the overlay:

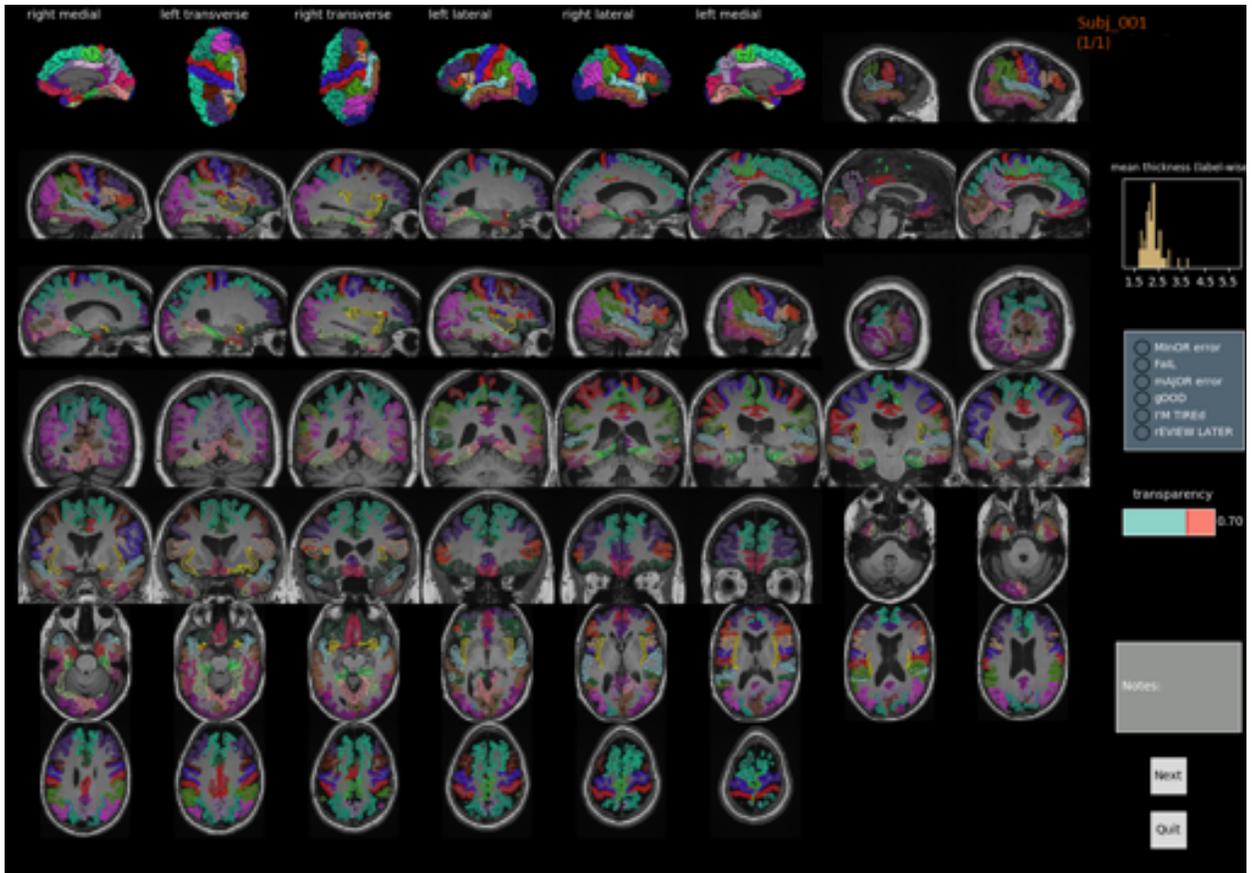


3.3.4 Transparency of the overlay can also be adjusted by dragging the green bar under 'transparency', for better contrast.



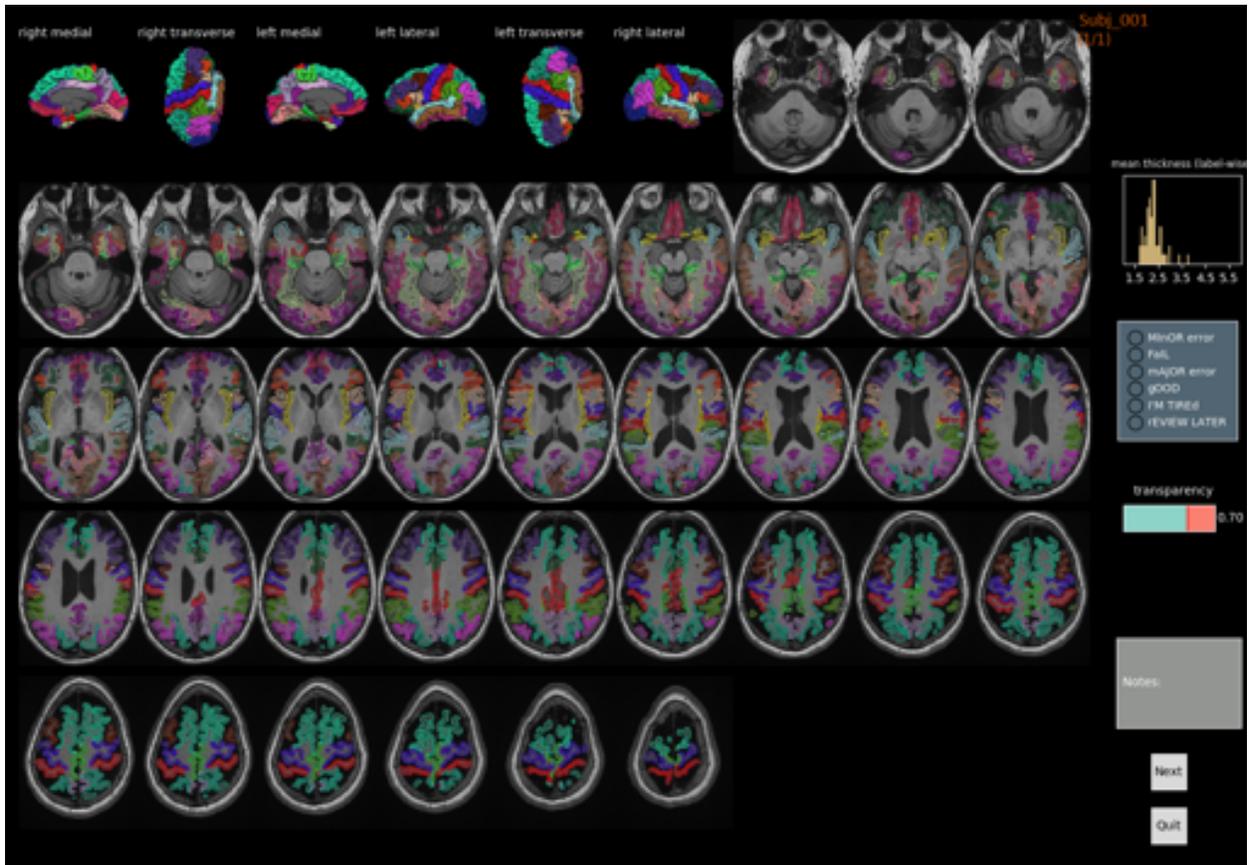
3.3.5 You can also adjust the default views or number of slices in the interface, using the -w and -s flags, respectively. For example, if you are worried about missing potential errors with the default 12 slices, you can increase this number to something like 16, by running the following command:

```
vcqfs -f </path/to/freesurfer/output/> -s 16
```



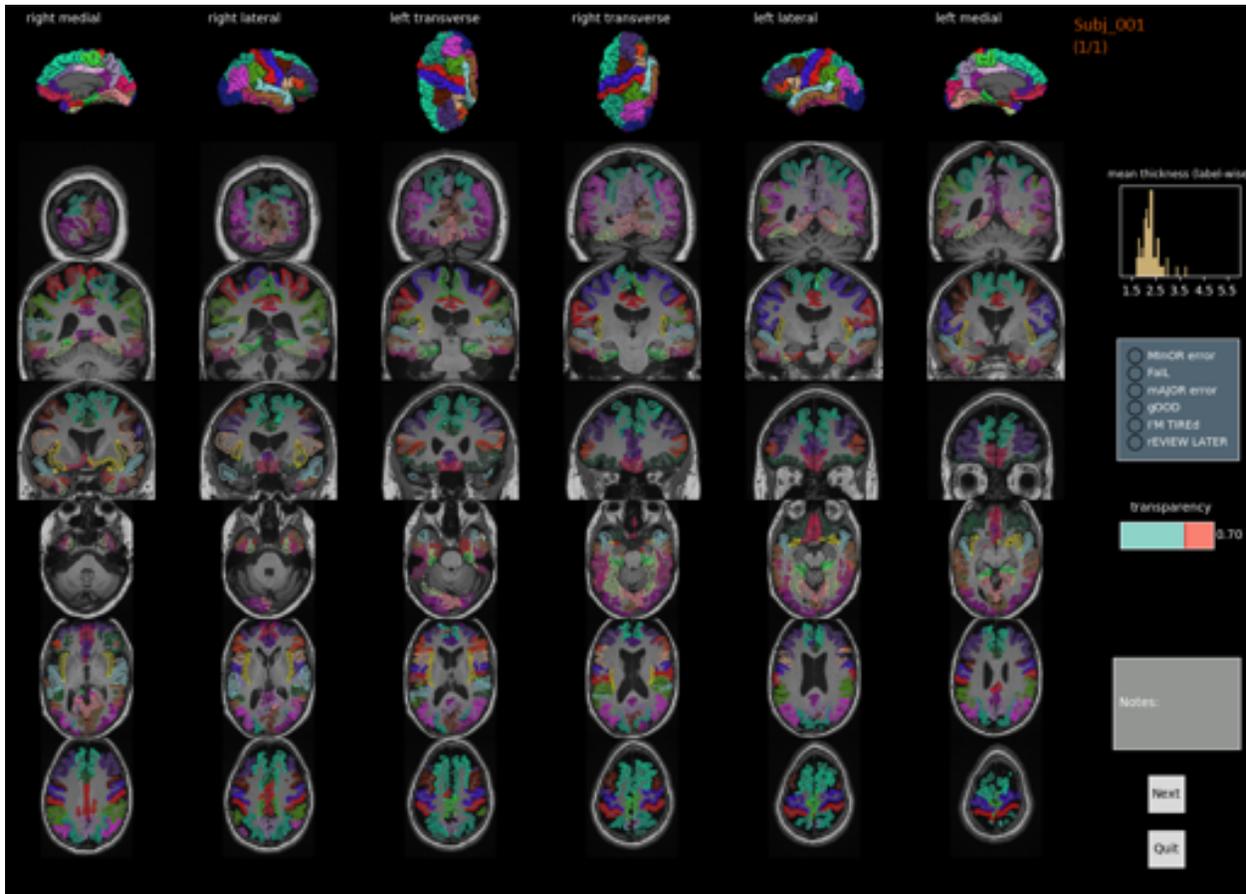
- 3.3.6 If you find it more helpful to look at more slices but only one or two different views, you can specify which views to look at using the `-w` flag (In most cases, 0=sagittal, 1=coronal, 2=axial, however, these values may be swapped, depending on your system. To test, you can try opening VisualQC for a single subject to figure out the orientation for your computer). You may also wish to adjust the number of rows for a better distribution of the images. For example, if we want to have 36 axial slices set into 4 rows, we would type:

```
vqcfs -f </path/to/freesurfer/output/> -s 36 -w 2 -r 4
```



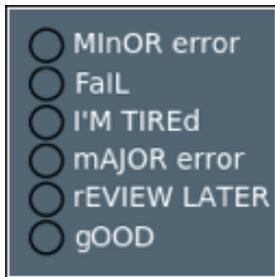
Or if we want the same number of views, but now split between axial and coronal (18 each) we can run this instead:

```
vqcfs -f </path/to/freesurfer/output/> -s 18 -w 1 2 -r 3
```



Or any other combination that you choose. Feel free to play around until you find a visualization that works best for you.

3.3.7 Error ratings are made using the radio buttons below. Selections can be made, either by selecting the button manually or by typing the lower case letter within the first word of the selection, e.g. 'g' for gOOD, 'm' for mAJOR error, 'r' for rEVIEW LATER, etc.



The ratings are as follows: Good – scan with no observable segmentation errors; Minor Error – only observed errors are minor ones, e.g. potential ROI misclassification (see section 3.4 and Appendix A for more details and examples); Major Error – one or more noticeable errors (e.g. pial over/underestimate) detected in scan; Fail – global segmentation failure; I'm tired/Review later – temporarily skip scan, no current rating

3.3.8 The Notes section is where you can type additional notes to record the exact errors detected in the scans. You can put whatever you like here, although it is recommended to keep notes short, otherwise it may take awhile for the text to load. One method is to note which ROIs the error appear in; suggested acronyms can be found in the Appendix A, to reduce the length of your notes. Make sure you are selected on the Notes section before typing; otherwise VisualQC may take certain keystrokes as commands to either change ratings or toggle the overlay on and off.

Note: The visible Notes section is very small, so you may not be able to see all of your notes as you type. They will still be saved to the .csv file, so long as you give them enough time to be recorded. If you can no longer see the end of the line you are typing, wait until the letters stop flashing between bold and regular font to indicate that all current notes have been recorded.

3.3.9 To advance scans, press the Next button. To temporarily quit, press Quit or ctrl-Q. Do not press the 'x' at the top! Your ratings will not be saved if you do.

Note: you cannot advance without giving a rating for the scan, so if you want to skip or exit without reviewing the current scan, make sure to select Review later or I'm tired before leaving. Then when you are ready to start reviewing scans again, delete the entries with Review later/I'm tired in them and save before re-opening VisualQC, so that you can start back where you left off.

3.3.10 As a default, VisualQC makes a new directory called visualqc, within the directory where your FreeSurfer data is stored. This is where the external visualizations and ratings files will be stored. Your ratings will be saved to a .csv file, within the 'ratings' subfolder.

3.4 Common Errors and Rating Scale

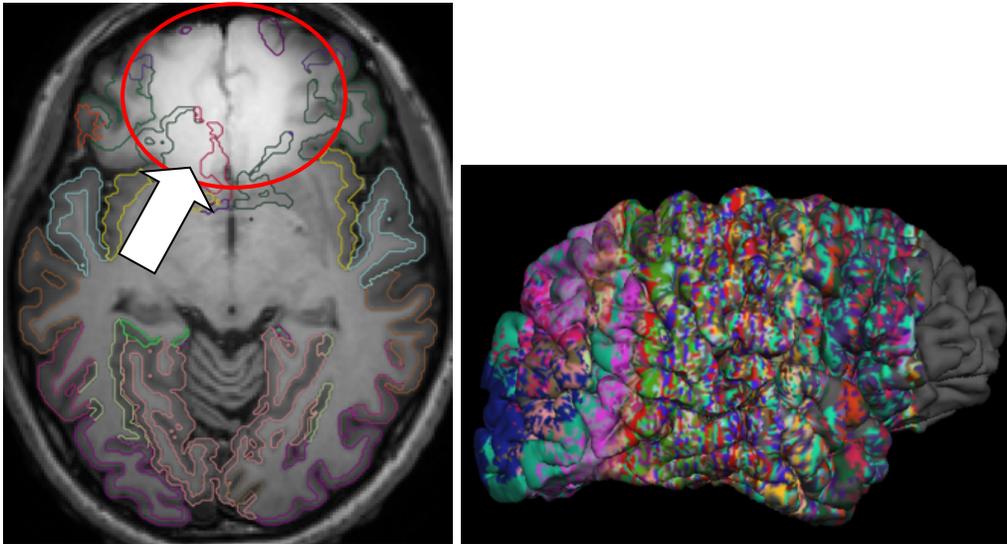
3.4.1 FreeSurfer output errors can be broken up into four main categories: Global failure, pial overestimates, pial overestimates, and ROI misclassifications. This section will briefly go into a description of each of these errors and a recommendation as how to rate them. Aside from global failure, all errors can occur in combination with each other and should be recorded as such in the Notes section.

Main errors at a glance:

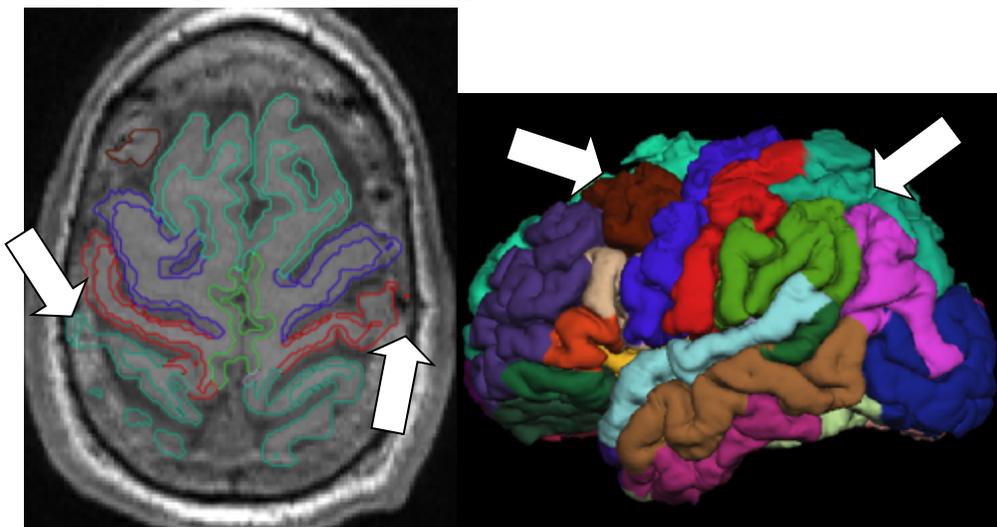
VisualQC User Manual

Error	Common ROI(s)	Severity	Frequency of Error
Global Segmentation Error	N/A	Fail	<1%
Pial Overestimate	postcentral, precentral, superior parietal	Moderate	~30%
Pial Underestimate	temporal pole, superior temporal, inferior temporal	Moderate	~35%
ROI Misclassification	banks of superior temporal sulcus	Moderate	~25%
	pericalcarine, lingual, cuneus	Moderate	25-30%
	insula	Minor	~30%
	entorhinal cortex, parahippocampal	Minor	80-100%
	cingulate	Minor	5-10%

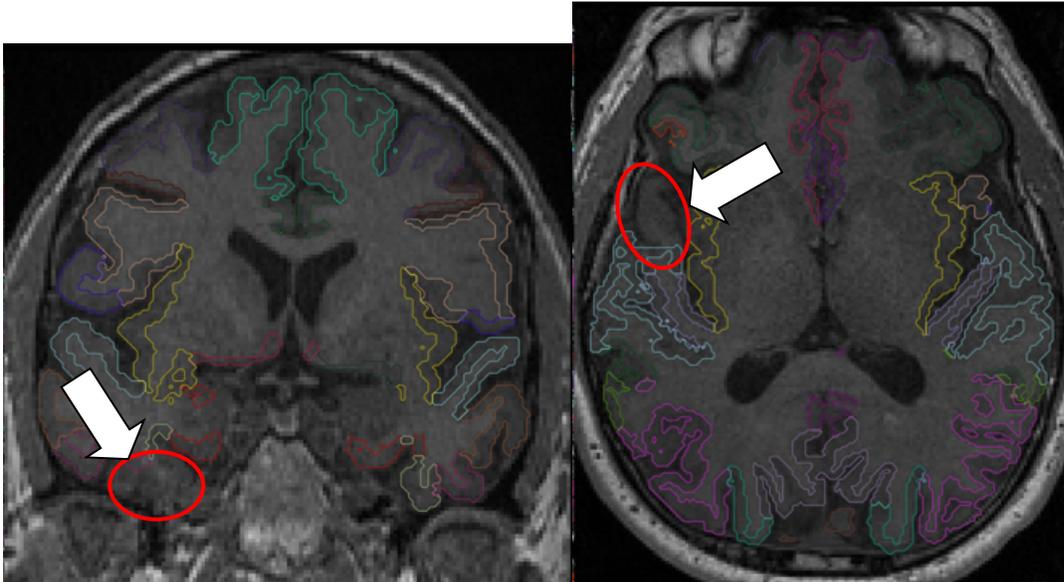
3.4.2 **Global segmentation error:** this occurs when a large portion of the brain has been excluded from the segmentation, usually due to pathology or an imaging artifact. (Note: This is only an error if FreeSurfer has failed to include a large segment of tissue that is in the scan. When the tissue is truly absent due to atrophy or abnormal anatomy, this is NOT an error) Another way this can occur is if there is a processing error that fails to produce a sensible segmentation map. When this occurs, the entire segmentation is considered to have failed and should be marked as 'Fail' under the error ratings, with 'Fail' written in the Notes section.



3.4.3 **Pial overestimate:** this occurs when the pial surface generated by FreeSurfer, extends beyond the grey matter surface and into the dura or skull. In this case, the scan should be marked with 'Major Error' in the ratings box and a description added to the Notes section that denotes the error and its rough location. E.g. 'pial overest. top RL' for a pial overestimate in the superior slices in both hemispheres. Optional: adding a ranking for the error based on severity, e.g. minor for a slight overestimate, moderate for a larger overestimate and severe for overestimates that extend well beyond the true pial surface or cover a large section of the brain. If the error is small, make sure that it is truly an error by looking at the same point in multiple views (axial, coronal, sagittal) and toggling the outline on and off, as sometimes what can appear to be an error is simply due to the limitations of a 2D slice. If still uncertain, make sure to add a note in that regards before moving on.



- 3.4.4 **Pial underestimate:** this occurs when the pial surface generated by FreeSurfer, excludes some of the grey and/or white matter. While this can occur anywhere, this most frequently occurs in the temporal lobes. Similar to overestimates, this should be marked as 'Major Error' and a relevant description added to Notes. E.g. 'pial underest. temp. lobes L'.



Note: be careful to ensure that this is truly an underestimate and not just the temporal pole, as this ROI is outlined in grey and can be difficult to see against the T1 image, especially if your screen has low contrast. Changing transparency or toggling the segmentation on/off using the 't' key on your keyboard may help.

- 3.4.5 **ROI misclassification:** in some cases, parts of the cortex can be mislabelled as a different ROI, resulting in an overestimate of the incorrect region and an underestimate of the correct one. Caution should be taken when making these ratings as this can be subjective and variations in anatomy can make an ROI appear to overestimated/underestimated, when it has been segmented correctly. If you're looking for a guide on which of these should be considered misclassifications, you can refer to the ENIGMA 2.0 protocol at: <http://enigma.ini.usc.edu/protocols/imaging-protocols/> or read the notes in the Appendix A of this manual. Depending on the misclassification and the ROIs involved, this can be considered a major or a minor error (details in Appendix A) Note: these areas only affect analyses that depend on correct segmentation of these particular regions. If looking at global measures or at ROI(s) distant from these point(s), these errors can be ignored.

3.5 Troubleshooting

The following are common errors or difficulties that can arise while using VisualQC, along with possible solutions.

VisualQC User Manual

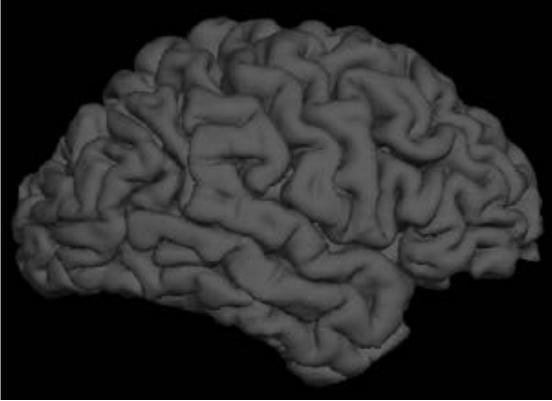
- 3.5.1 In case you accidentally close VisualQC without saving, or an error occurs while saving ratings, know that VisualQC outputs the ratings for each session into the command line as you go through them, which can be used to recover previous ratings. This will be automatic in the future.

Note: it is also recommended that you save a backup of your current ratings before reopening VisualQC, if you are completing ratings in multiple blocks. This can save some headache, especially when rating a large number of subjects or performing the review on a laptop with dying battery etc..

- 3.5.2 VisualQC doesn't reload participants marked 'Review Later' or 'I'M tired'. This is a current bug with the interface but a simple workaround is to open the ratings .csv file and delete all participants with that rating and saving. Then the next time VisualQC is opened, these participants will be reloaded into the viewer. Note: this can also be used to fix or update an incorrect rating, or to delete participants that you want to review again, regardless of their previous rating.
- 3.5.3 If you receive the following error: `ValueError: Invalid fraction of outliers: must be more than 1/n (to enable detection of at least 1)`, it means that VisualQC was unable to complete outlier detection, probably due to too few participants being analyzed. Outlier detection can be turned off by adding the flag '-old', after which it should run.
- 3.5.4 If you receive this message and no 3D surface visualizations appear, this means that VisualQC was unable to produce the surface visualizations, because it could not access a working version of FreeSurfer TkSurfer. If you do not have TkSurfer, or if you want to skip this option for whatever reason, you can shut off surface visualizations using the '-ns' flag.



- 3.5.5 In some cases, VisualQC does produce surface visualizations, but they lack the proper colour-coded ROI segmentation, like in the image below:



This occurs when VisualQC cannot find all of the necessary files to make the appropriate label; make sure that all of the correct label files are in your participants recon-all output file (complete list is available on FreeSurfer website here: <https://surfer.nmr.mgh.harvard.edu/fswiki/ReconAllDevTable>)

- 3.5.6 Typed notes in the ratings file are incomplete or carry on into the notes for the next participant. This occurs when you quit or hit next before the interface has a chance to load all of your keystrokes. To avoid this, try to keep notes short and make sure text has stopped appearing in the Notes section before you click quit or next (the letters will flash between bold and regular text while your keystrokes are still being recorded).

3.6 Additional Resources

If you run into any bugs, or have any suggestions on how to improve the VisualQC interface, feel free to open an issue at: <https://github.com/raamana/visualqc/issues>. Feedback is welcome. For additional documentation on VisualQC, feel free to visit: <https://raamana.github.io/visualqc/readme.html>

For an additional resource on rating FreeSurfer recon-all output, you can use the ENIGMA 2.0 Protocol: <http://enigma.ini.usc.edu/protocols/imaging-protocols/>, developed by the University of South California.

4.0 REFERENCES

Pradeep Reddy Raamana. VisualQC: Assistive tools for easy and rigorous quality control of neuroimaging data. <http://doi.org/10.5281/zenodo.1211365>

USC Mark and Mary Stevens Neuroimaging and Informatics Institute. ENIGMA [Internet]. Available from: <http://enigma.ini.usc.edu/>

Chris Ching, Faisal Rashid, Sophia Thomopoulos, Lianne Schmaal, Premika Boedhoe, Paul Thompson, and Neda Jahanshad. ENIGMA Cortical QC 2.0. April 2017.

5.0 REVISION HISTORY

SOP Code	Implementation Date	Summary of Changes
SOP_00x_00x.0x	13.Feb.2019	Initial draft.
v1.1	20.Feb.2019	Move ENIGMA QC to Appendix, add troubleshooting section
v1.2	30.Aug.2019	Add clarification for major vs minor errors, more detailed definitions for error classes
v1.3	15.Oct.2019	Add clarification for installation instructions, with additional troubleshooting, update images with new rating interface
v1.4	29.Nov.2019	Add section for vqcdeface in Appendix B

AUTHORS

- Athena Theyers
- Pradeep Reddy Raamana

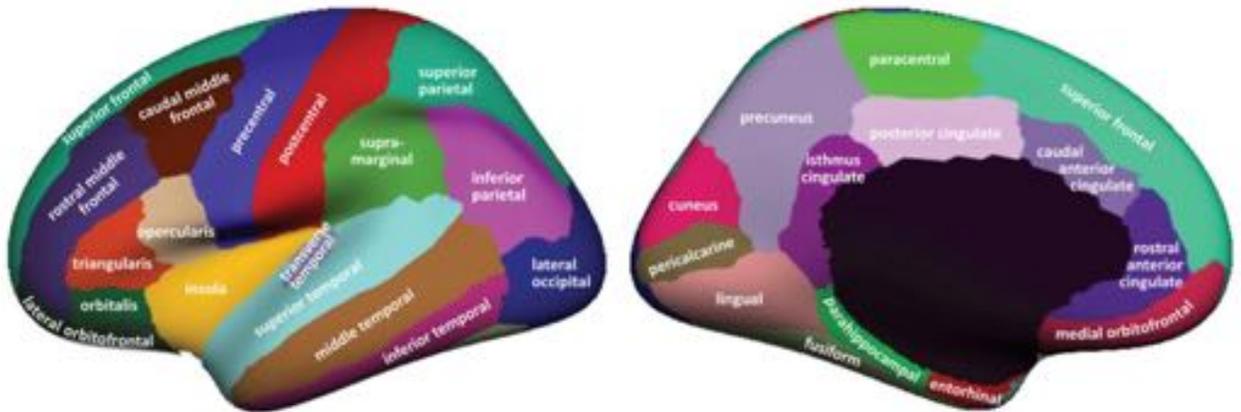
APPENDIX A: Comparison to ENIGMA QC 2.0

In order to compare results to another rating system, it is recommended to use short forms and codes that can be easily correlated to that other system. The following system outlines recommended short forms and note styles to compare to the ENIGMA QC protocol 2.0 <http://enigma.ini.usc.edu/protocols/imaging-protocols/>, along with an aid for counting errors from the ratings setting. An excel template is available, to count the number of errors by ROI and hemisphere, for your convenience.

Error categorization based on the ENIGMA Cortical QC protocol v2.0, should be recorded by hemisphere(s) and ROI(s) they were discovered in, separated by semicolons. The exception to this is for global segmentation errors, where the entire segmentation is considered to have failed, so a simple 'Fail' can be recorded in the Notes.

5.1.1 ROIs are defined by the Desikan-Killiany atlas, shown below. Suggested short forms are recorded in the table below, for recording in the Notes section. If other acronyms are easier for you to remember, feel free to change them, however, remain consistent and remember to update the attached excel template, accordingly, so that your error ratings are counted accurately.

NOTE: make sure to choose acronyms that are as unique and as distinct as possible. For example, acronyms xyz and yz for two different ROIs could be problematic, as both xyz and yz would be counted as an error in yz.

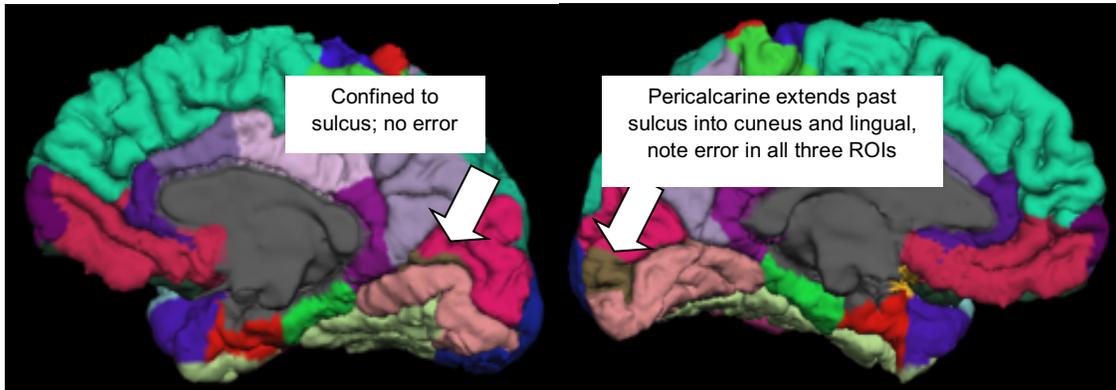


VisualQC User Manual

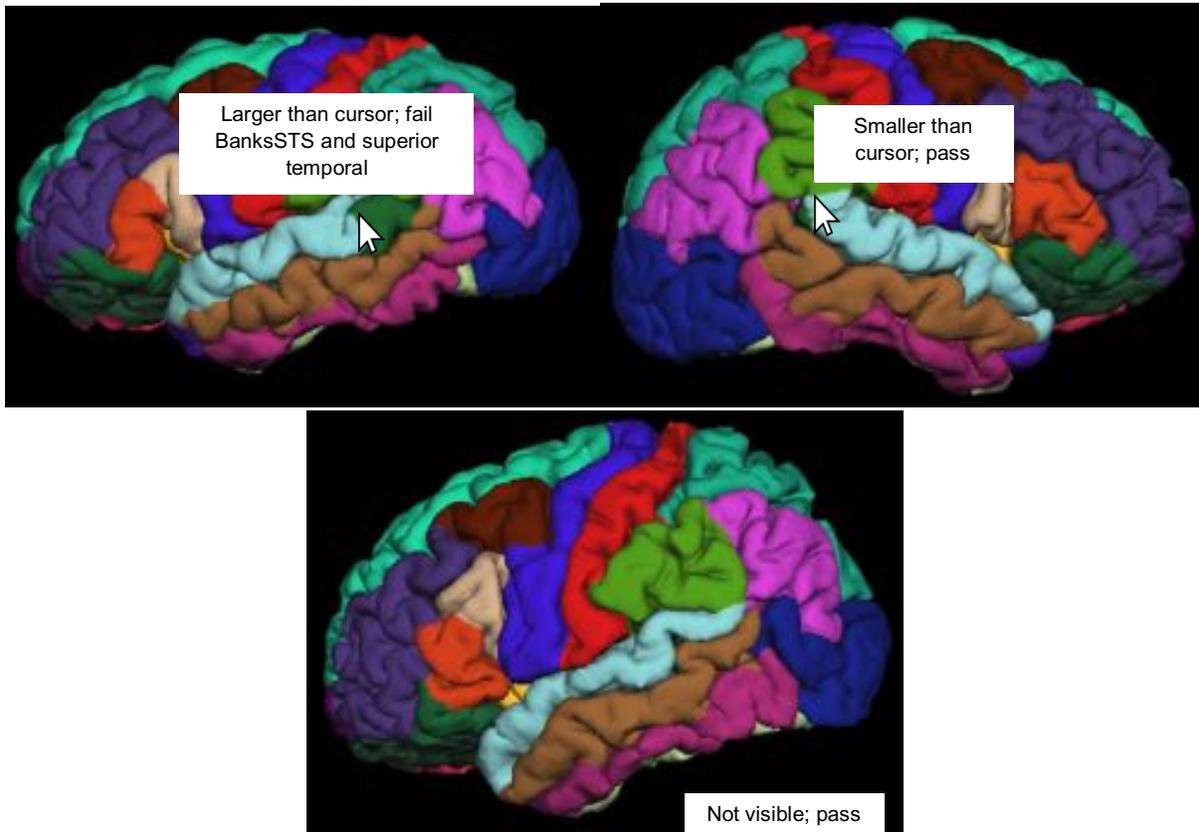
ROI	Acronym
banks of superior temporal sulcus	bssts
caudal anterior cingulate	cac
caudal middle frontal	cmf
cuneus	cu
entorhinal cortex	ec
frontal pole	fp
fusiform	fu
inferior parietal	ip
inferior temporal	it
insula	in
isthmus cingulate	ic
lateral occipital	lo
lateral orbitofrontal	lof
lingual	li
medial orbitofrontal	mof
middle temporal	mt
paracentral	pac
parahippocampal	ph
parsopectoralis	pop
parorbitalis	po
parstriangularis	pt
pericalcarine	pc
postcentral gyrus	pocg
posterior cingulate	poc
precentral gyrus	pcg
precuneus	pcn
rostral anterior cingulate	rac
rostral middle frontal	rmf
superior frontal	sf
superior parietal	sp
superior temporal	st
supramarginal	sm
temporal pole	tp
transverse temporal	tt

5.1.2 **Global Segmentation Error (Fail)** – All ROIs are considered to have failed in this case and should be recorded as ‘Fail’ in the Notes.

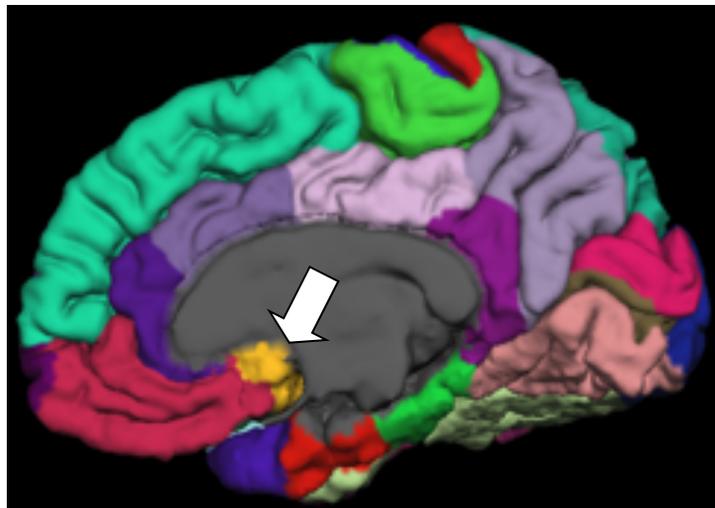
- 5.1.3 **Pial Overestimate/Underestimate (Major)** – In this case, all ROIs where this occurs should be recorded, along with L, R, or RL to indicate which hemisphere(s) this error occurred in.
- 5.1.4 **ROI Misclassification (Major): Pericalcarine overestimate** – the pericalcarine should be confined to the calcarine sulcus; when the pericalcarine extends significantly beyond this, it and any adjacent ROIs affected by this overlap (generally cuneus and/or lingual), should be marked as erroneous.



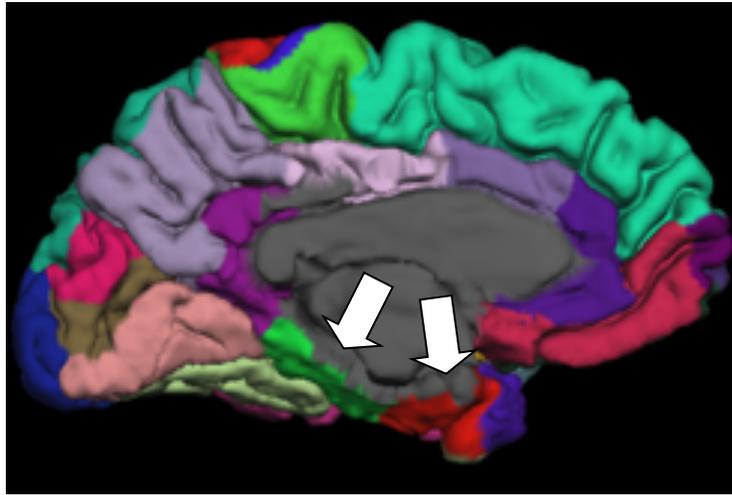
- 5.1.5 **ROI Misclassification (Major): Banks of superior temporal sulcus (BanksSTS) overestimate** – similar to the pericalcarine, the BanksSTS should be confined to the sulcus and not visible from the external view. When viewing the external view at about the same size as the brains below, the BanksSTS ROI, if visible, should be smaller than your cursor. (Equivalently, the BanksSTS should be smaller than this shape:  in the expanded view.) If this is not the case, the BanksSTS should be marked as an error, along with any ROIs it extends into (usually superior temporal and/or middle temporal)



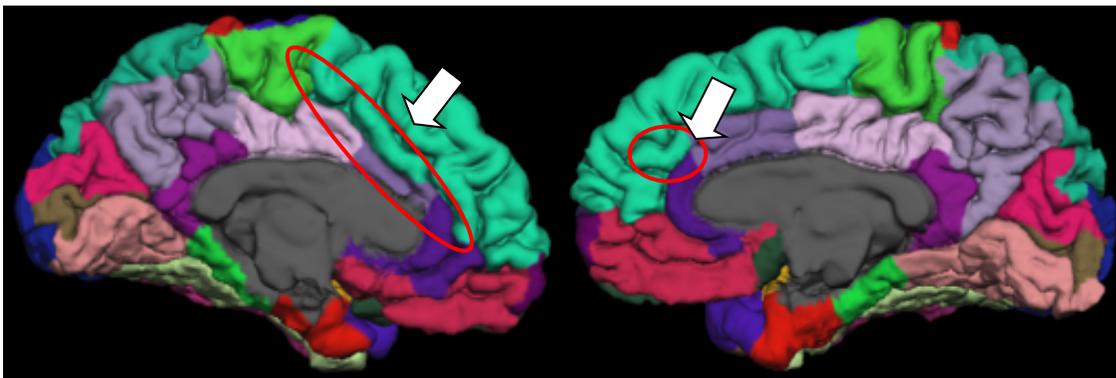
5.1.6 **ROI Misclassification (Minor):** Insula overestimate – since FreeSurfer lacks a separate label for subgenual anterior cingulate cortex (ACC), this region is sometimes [mis-]labelled as the insula. While technically incorrect, the missing label and variability in anatomy make this a minor error.



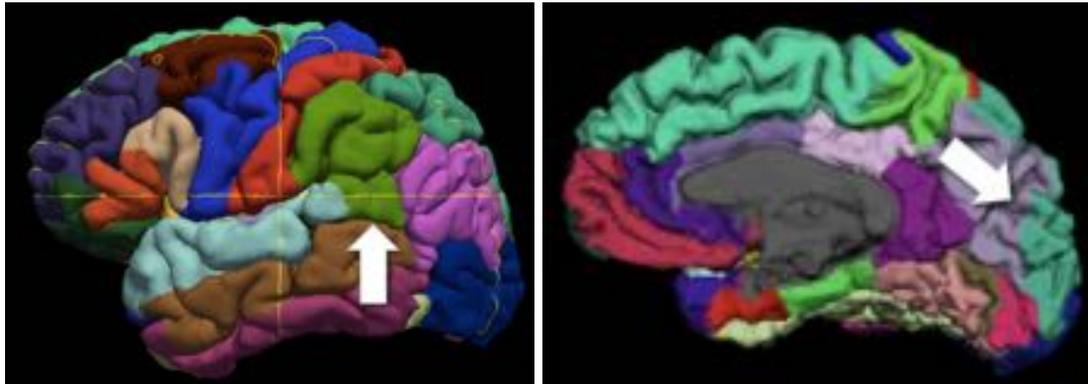
- 5.1.7 **ROI Misclassification (Minor):** Entorhinal cortex and parahippocampal underestimate – this area is routinely underestimated and is considered to be more minor of an error. A note should still be made so interpretations of stats in this area are made with caution, but this is not generally considered a segmentation failure of this region. Alternatively, you may wish to only note sessions where a large section of the entorhinal cortex or parahippocampal regions are underestimated; just be sure to remain consistent.



- 5.1.8 **ROI Misclassification (Minor):** Cingulate cortex – roughly 30-60% of the population have a paracingulate sulcus (extra fold between superior frontal gyrus and cingulate cortex) which can result in an underestimate of the cingulate cortex and an overestimate of the superior frontal gyrus. However, due to high variability in morphology of this region, it is difficult to be certain of ROI boundaries, so this error is generally considered minor.



5.1.9 ROI Misclassification (Minor): Superior parietal and supramarginal overestimate – in some cases, ROIs like the superior parietal lobule and the supramarginal gyrus extend past their normal regions - into the cuneus and/or precuneus for the superior parietal and into the temporal poles for the supramarginal. Due to anatomical variability, this is considered a minor error, as exact boundaries are difficult to define.



5.1.10 Once ratings are complete, the three columns produced in the ratings .csv file can be copied into the VisualQC excel template so a summary of the number of each error can be collected for the dataset. An example is shown below:

ROI	total errors	% of total	% in both	% in left H	% in right
bankssts	3	25.00%	8.33%	16.67%	0.00%
caudal/medialcingulate	1	8.33%	8.33%	0.00%	0.00%
frontalpole	2	16.67%	16.67%	0.00%	0.00%
granular	3	25.00%	8.33%	8.33%	8.33%
entorhinal	4	33.33%	25.00%	8.33%	0.00%
frontalpole	1	8.33%	8.33%	0.00%	0.00%
pericalcarine	1	8.33%	8.33%	0.00%	0.00%
isthmuscingulate	2	16.67%	16.67%	0.00%	0.00%
insula	1	8.33%	8.33%	0.00%	0.00%
isthmuscingulate	1	8.33%	8.33%	0.00%	0.00%

Subj	ROI	Error
Subj_001	gOOD	No Errors
Subj_002	nor en in:R	Only Minor Errors
Subj_003	fail	Global Segmentation Error
Subj_004	maJOR er cmf:RL;rmf:R	Pial Overestimate
Subj_005	maJOR er bstst:l;st:l;nt:l	BanksSTS Overestimate
Subj_006	maJOR er ec:RL;ph:L	
Subj_007	maJOR er sp:R;ip:RL;ec:RL	
Subj_008	maJOR er pc:L;cu:L;li:L	Poles Underestimate
Subj_009	maJOR er tp:RL;st:R	
Subj_010	maJOR er bstst:l;st:L;pc:R;cu:R;li:R	
Subj_011	maJOR er in:R;ec:L	
Subj_012	maJOR er sf:R;pocg:R	

APPENDIX B: `visualqc_defacing`

Another useful tool available in the `visualqc` library, starting with v0.4, is a quality control interface for defaced mri scans. Due to a growing number of publicly available mri datasets and concerns around preserving participant privacy, it has become increasingly important to de-identify all data before release, including removing facial features from structural mri scans. While there are multiple freely available automated pipelines, the only way to be certain that 1) the face has successfully been removed and 2) that no brain was removed, rendering the scan unusable, is to manually inspect the output. `Visualqc` provides a tool for this, called `visualqc_defacing`.

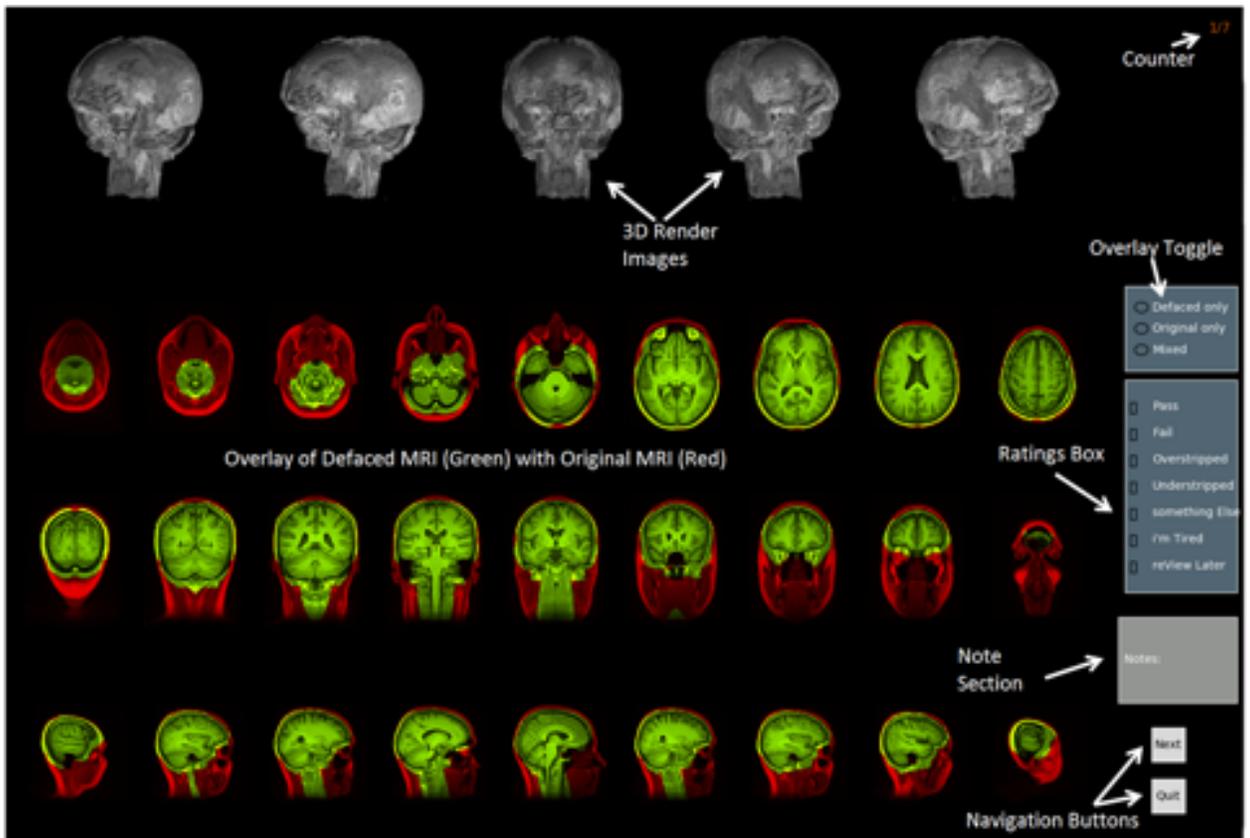
5.2 Running `visualqc_defacing`

- 5.2.1 To ensure that your version of `visualqc` has the defacing interface, try running either `visualqc_defacing -h` or `vqcdeface`. You should get the help message for `visualqc_defacing`. If you get an error instead, try running `pip install -U visualqc` to ensure you have the latest version of `visualqc` and try re-running. If this still fails, double check your `PATH` variable to ensure that the path to the `visualqc` program files is included and that you are running `python3`.
- 5.2.2 To run `vqcdeface` on a dataset, make sure that each subject has their own folder, containing the original, pre-defaced image, the defaced image(s), and png snapshot(s) of the 3D rendered defaced image. At a minimum, one snapshot of the front of the defaced scan is required, but it is recommended to have a couple of different angles, to capture a more complete image of the scan. All snapshots should have the same prefix to view them at the same time. Note: make sure the prefix is different from all of the mri files in the same folder, e.g. `defaced.png` and `defaced.nii.gz` will cause an error but `Render_deface.png` is fine. For more information on how to create a 3D rendered image of your defaced MRI, see Appendix C.
- 5.2.3 To open the interface, run `vqcdeface -u /path/to/dataset/ -d defaced_scan -m original_scan -r 3Drender_prefix`. To specify, only certain participants, create a text file with a list of the participants' names, similar as with `vqcfs` in section 3.2.2. The text file can then be submitted using `-i subj_names.txt`.

5.3 The Interface

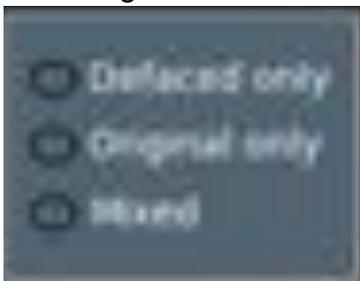
Disclaimer: all MRIs depicted in the following examples for this section and the next, are taken from average templates of multiple participants' scans, or have been masked to preserve privacy.

- 5.3.1 After `vqcdeface` has finished loading, you should get a screen like this:

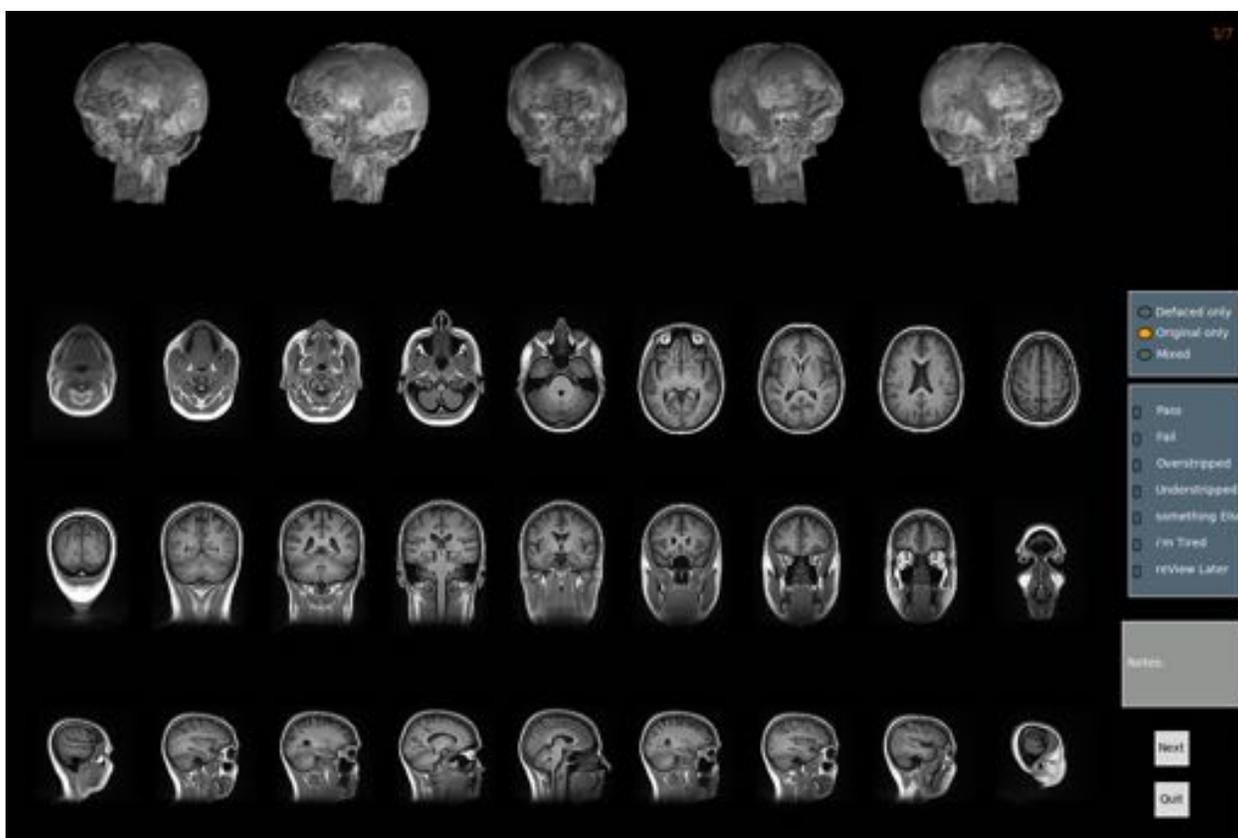
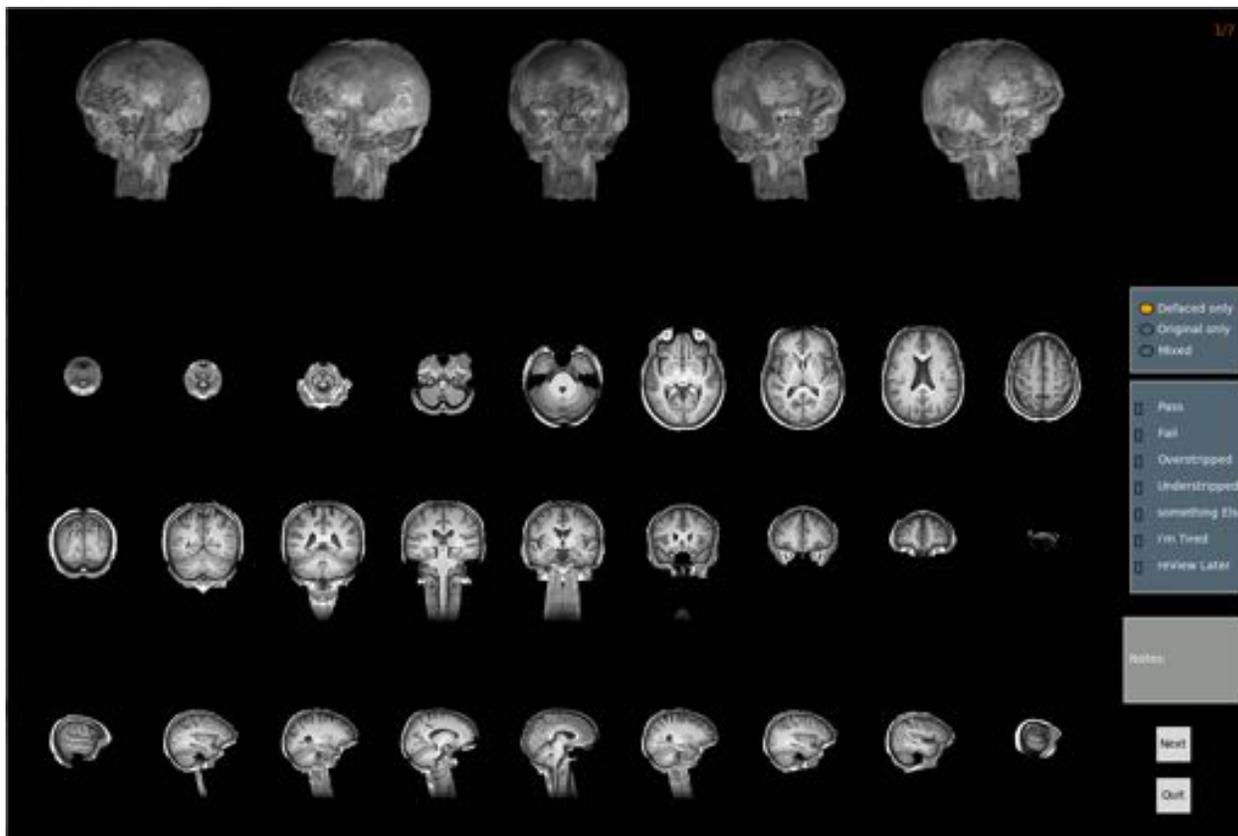


With the top row consisting of the 3D rendered images, and the bottom three contain an overlay of the defaced MRI in green, with the original MRI depicted in red. On the top right is a counter to keep track of the rater's position in the dataset. Note: participant IDs are no longer visible within the viewer, to reduce rater bias of repeated site or participant scans.

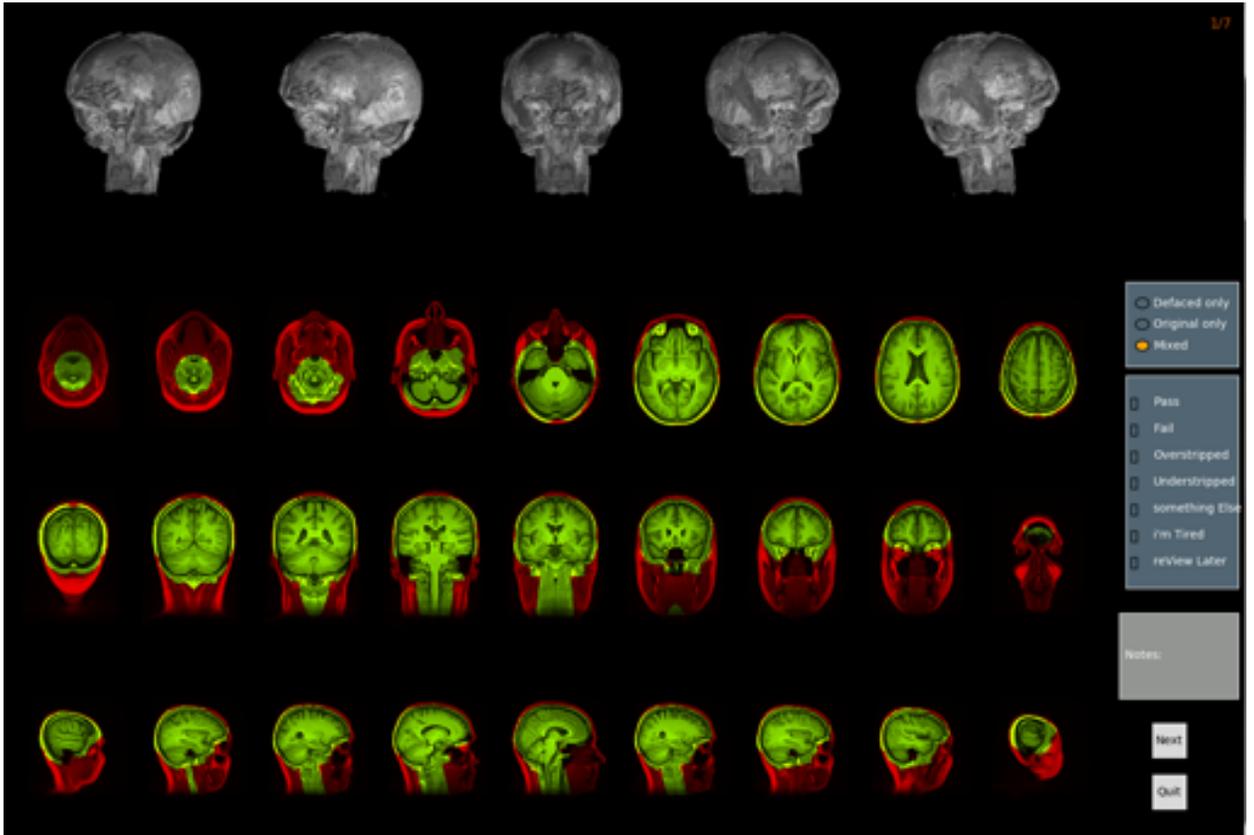
- 5.3.2 By default, the interface loads both the defaced and the original mri, but the rater can select to view just the original or the defaced image using the toggle buttons on the right:



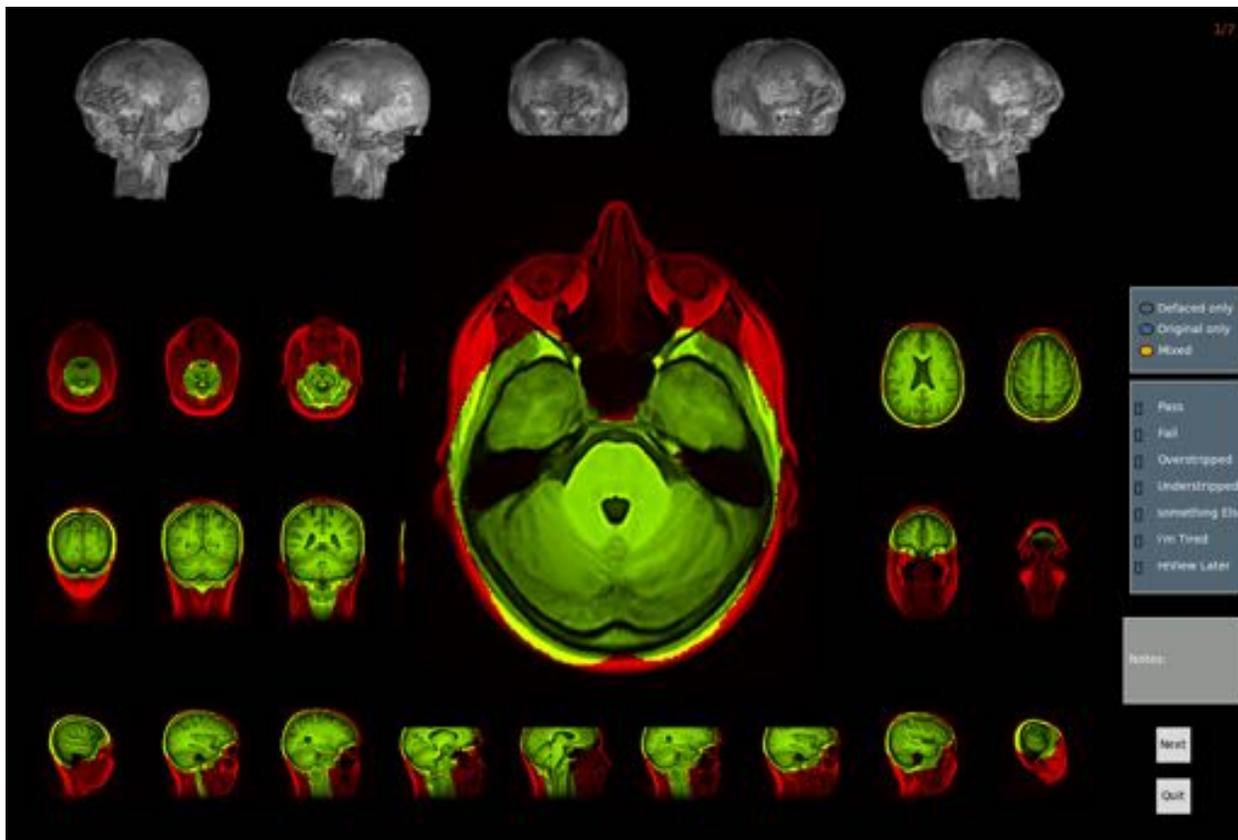
VisualQC User Manual

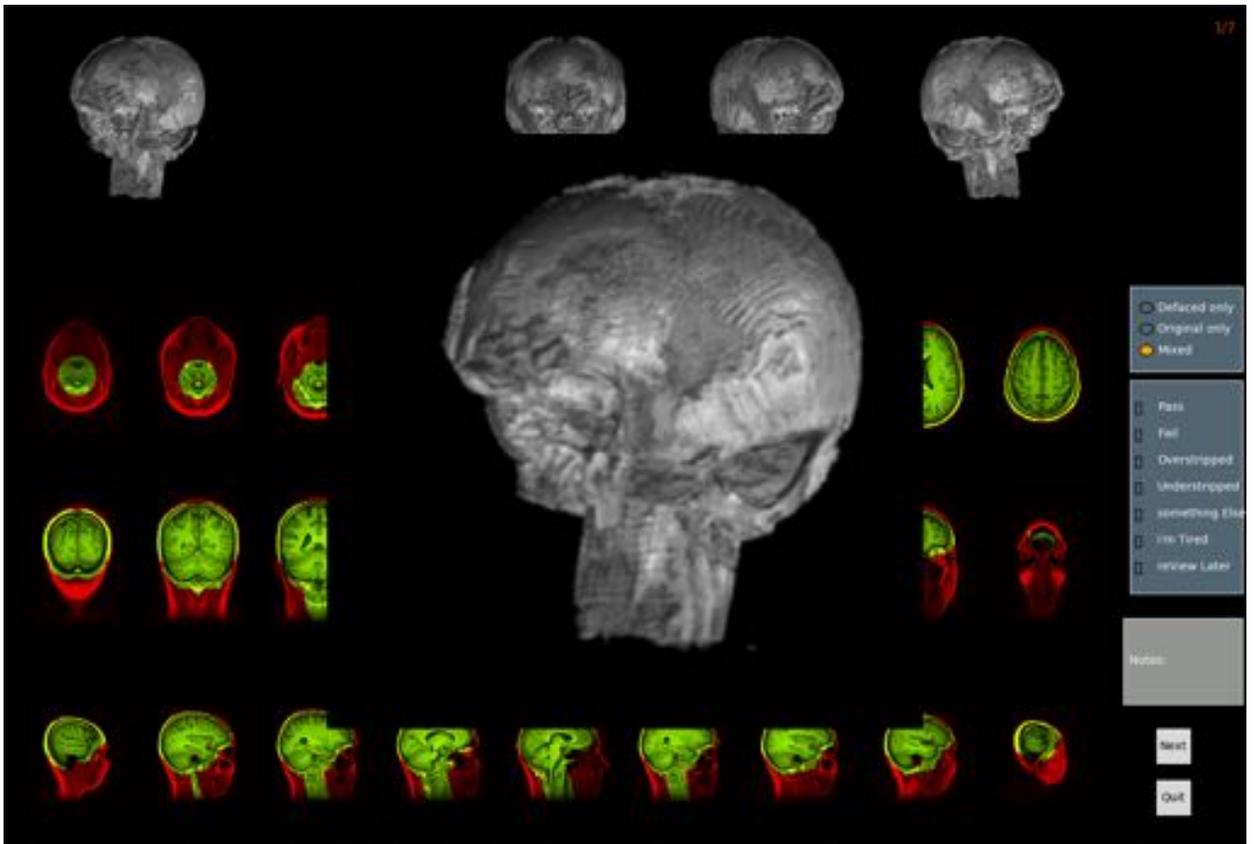


Then switch back using mixed



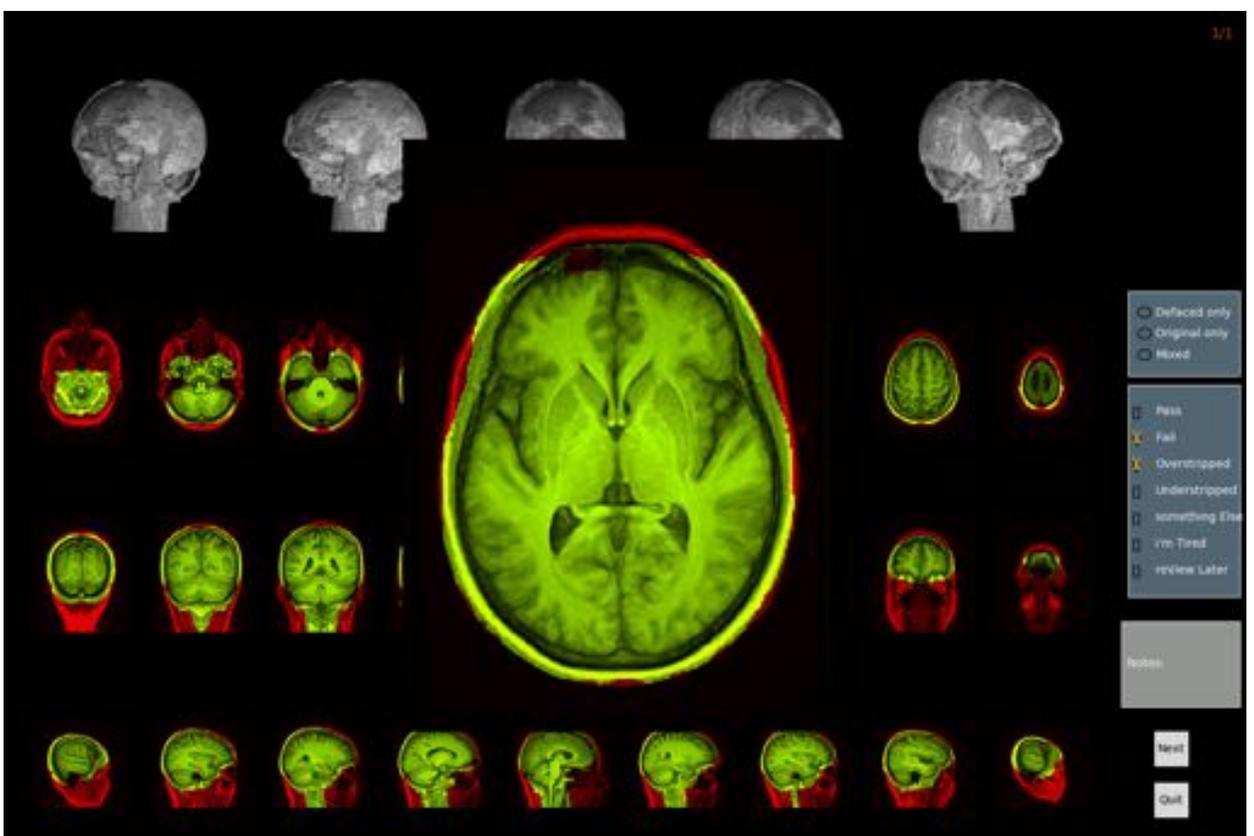
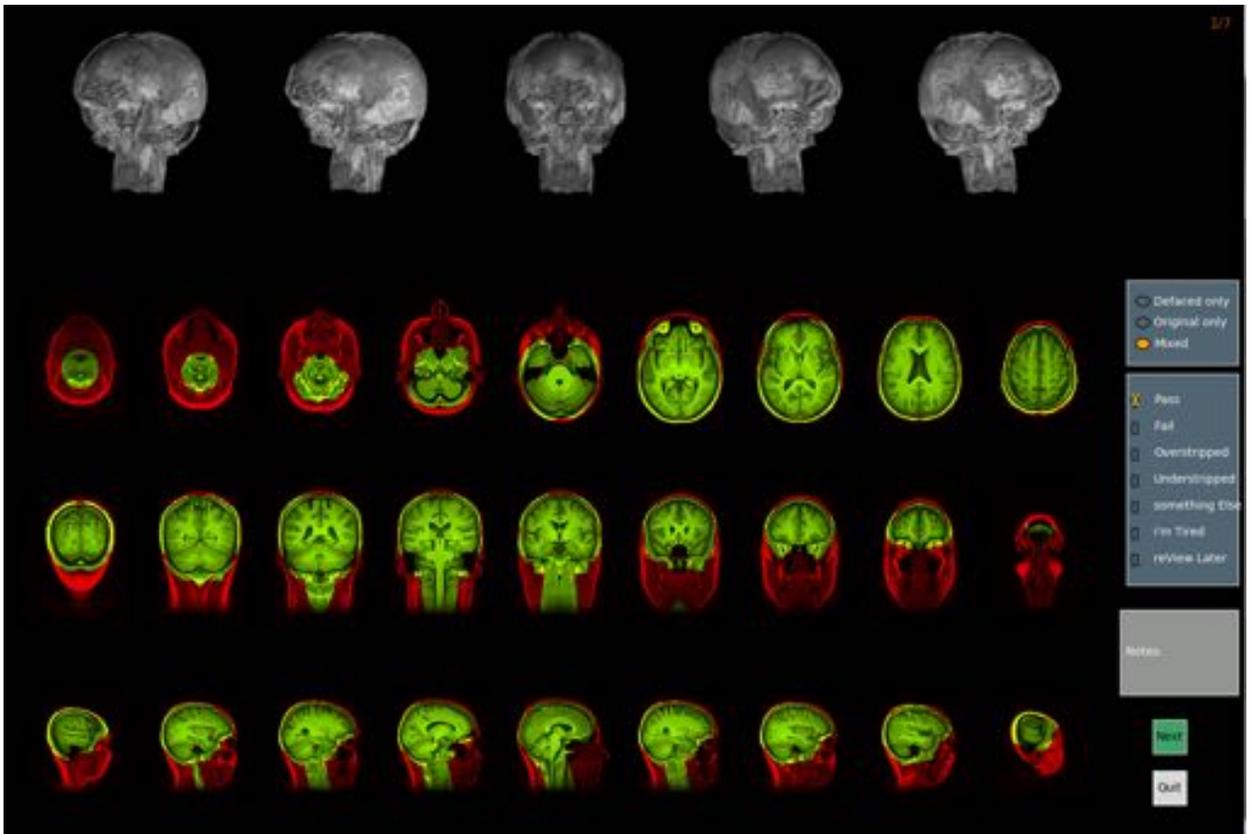
5.3.3 Like with vqcfs, individual slices and rendered images can be zoomed in on, by double clicking on the image. Clicking again, will reduce the image again.



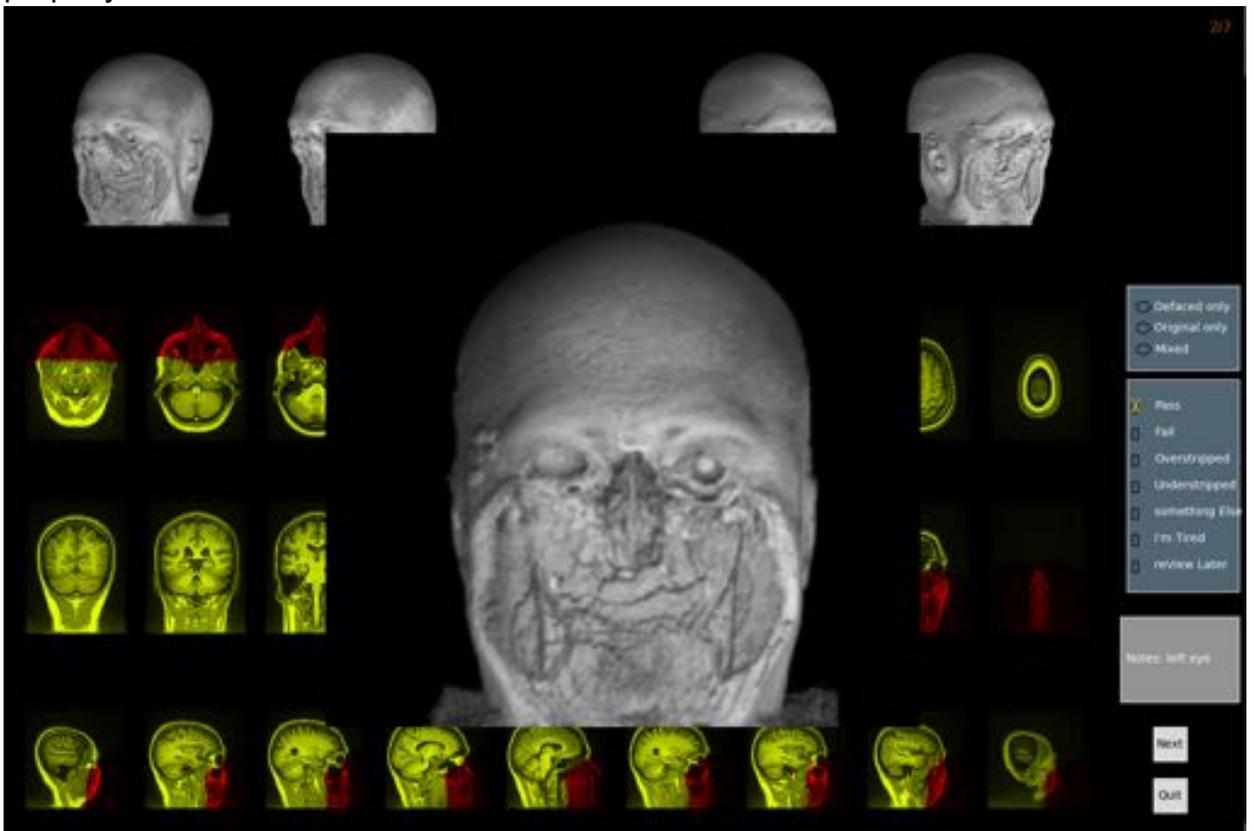


5.3.4 Ratings can be made by selecting either 'pass' in the rating box or 'fail', then 'overstripped', i.e. some brain has been removed and/or 'understripped', i.e. face or partial face remains in the 3D render. Note: you must select a rating to either move onto the next scan or quit. To skip rating a scan, select 'review later' and hit either 'Next' to progress to the next scan, or 'Quit' to close the interface and save all generated ratings.

VisualQC User Manual



- 5.3.5 In many cases, it would be useful to expand on ratings, e.g. include which sections of the brain have been removed by the defacer and by how much, or which parts of the face still remain after the MRI was supposedly defaced. This can be done by typing a short description into the grey Notes box. This can also be used in cases where the scan may be passed, if there is something that might be important to future users of the scan. Note: make sure the letters have finished entering into the box (the text will stop flashing between bold and regular font if your note is long enough that the end progresses beyond the edge of the visible box) before moving to the next scan or your notes might not be properly recorded.



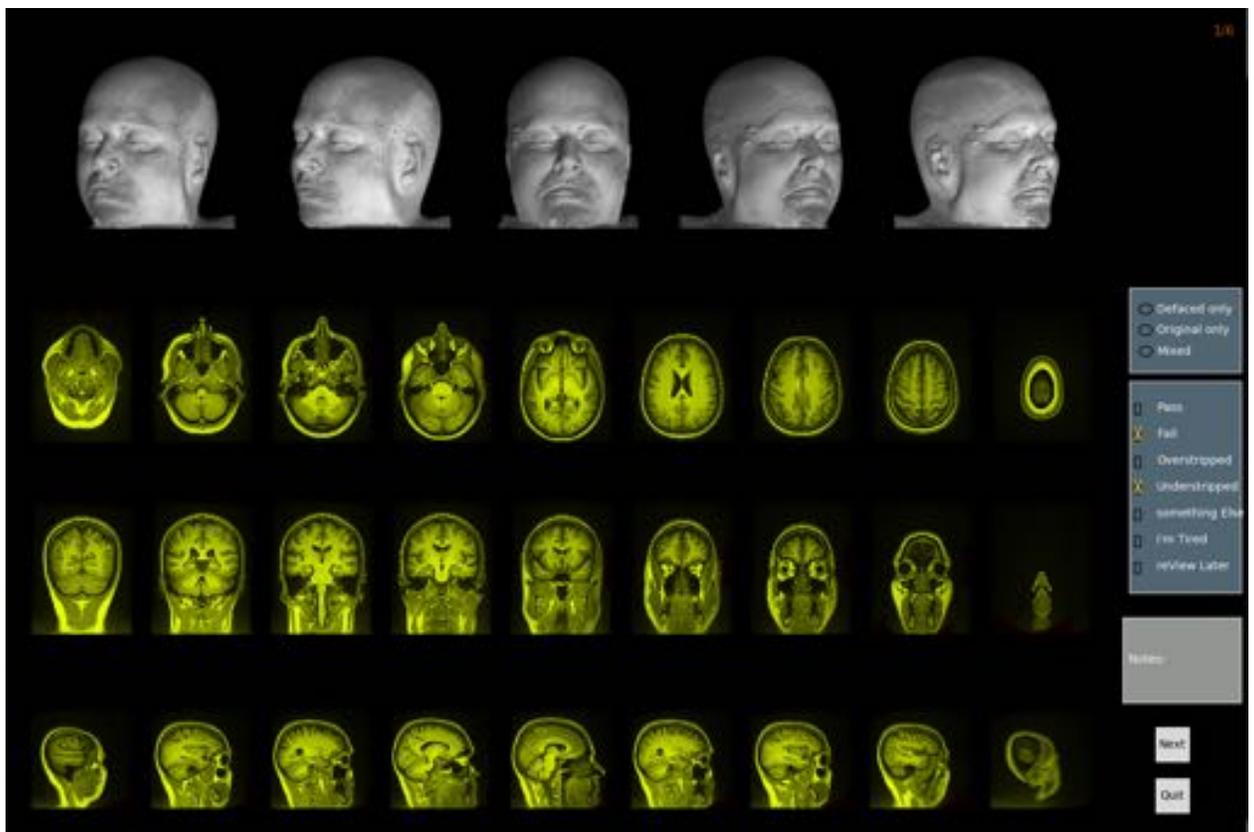
- 5.3.6 When exiting partway through the dataset, make sure to hit the 'Quit' button after rating your final scan, rather than the 'x' on top. If you don't, all of the ratings for your session will not be saved.
- 5.3.7 As a default, vqcdeface will create a folder called visualqc within your subject directory and save the ratings in a .csv file within a subfolder called ratings. This is the file that the interface will reference when you reopen vqcdeface to resume ratings. Be sure to delete any subjects with 'review later' ratings, or those that you simply need to go over again from the .csv file and save, so that the interface will reload them. To start over from the beginning, delete the .csv file

and `vcdeface` will reload all scans. To save files in a location other than the subject directory, use `-o /path/to/ratings` when opening the interface, and the 'visualqc' folder will be created within this directory.

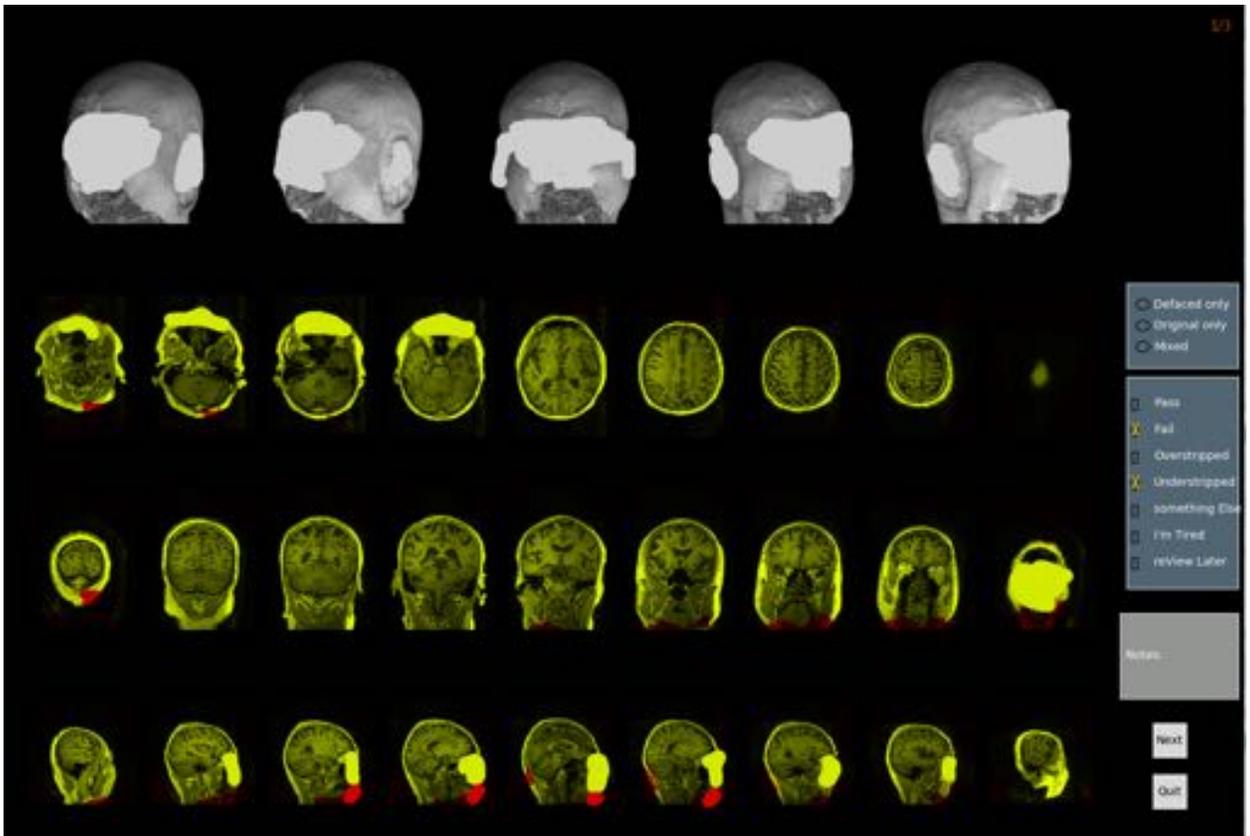
5.4 Common Defacing Errors

Below are some common errors that can be found when defacing MRI scans, as collected from several different defacing softwares

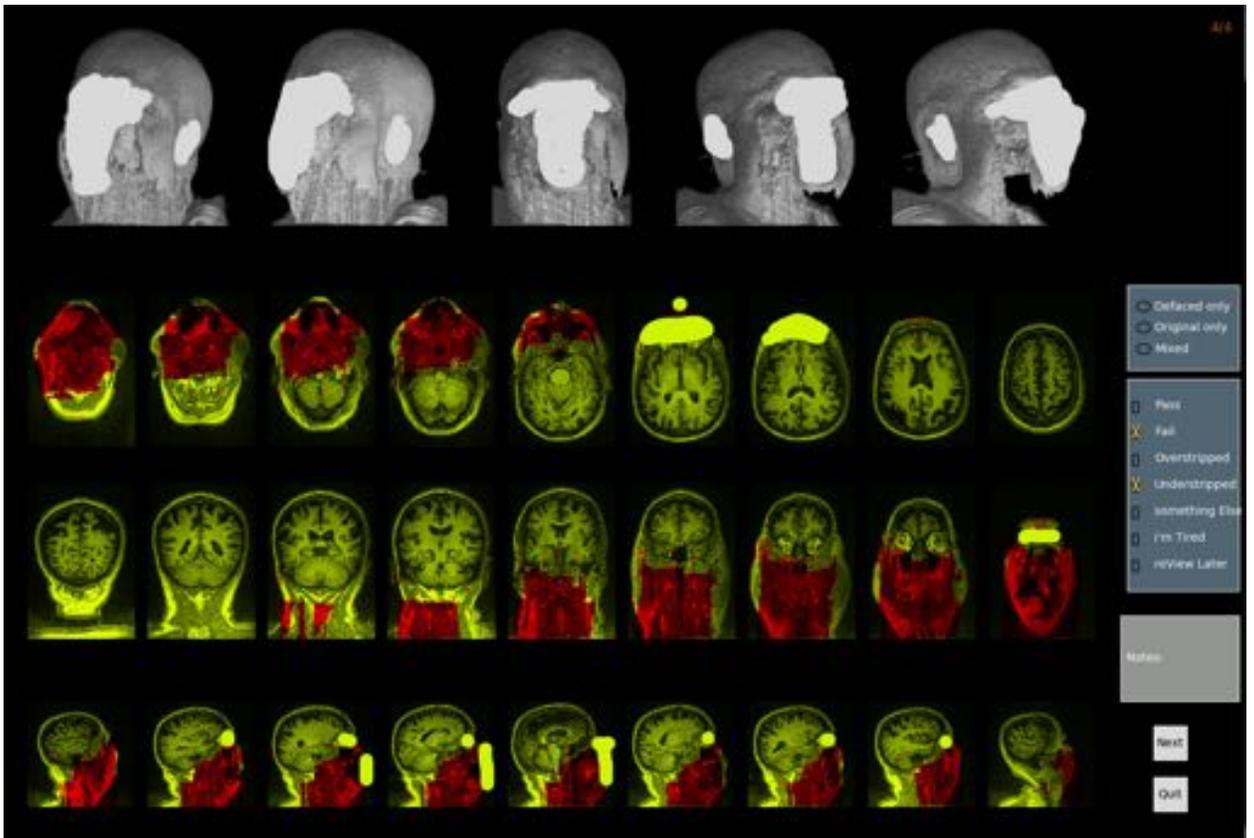
- 5.4.1 Complete failure to deface scan: in some cases, the defacing software can miss the face entirely, either removing only a small section of the background or leaving the scan completely untouched. This should be marked as 'Fail', and 'Understripped'



- 5.4.2 Partial defacing, but most of the face missed: this is the case where the defacer underestimates the size of the face, either removing only the bottom section of the face (e.g. mouth and chin, but leaving eyes and nose) or removing only one side of the face (e.g. masking the right half of the face, but leaving the left) This again should be marked as 'Fail', and 'Understripped'

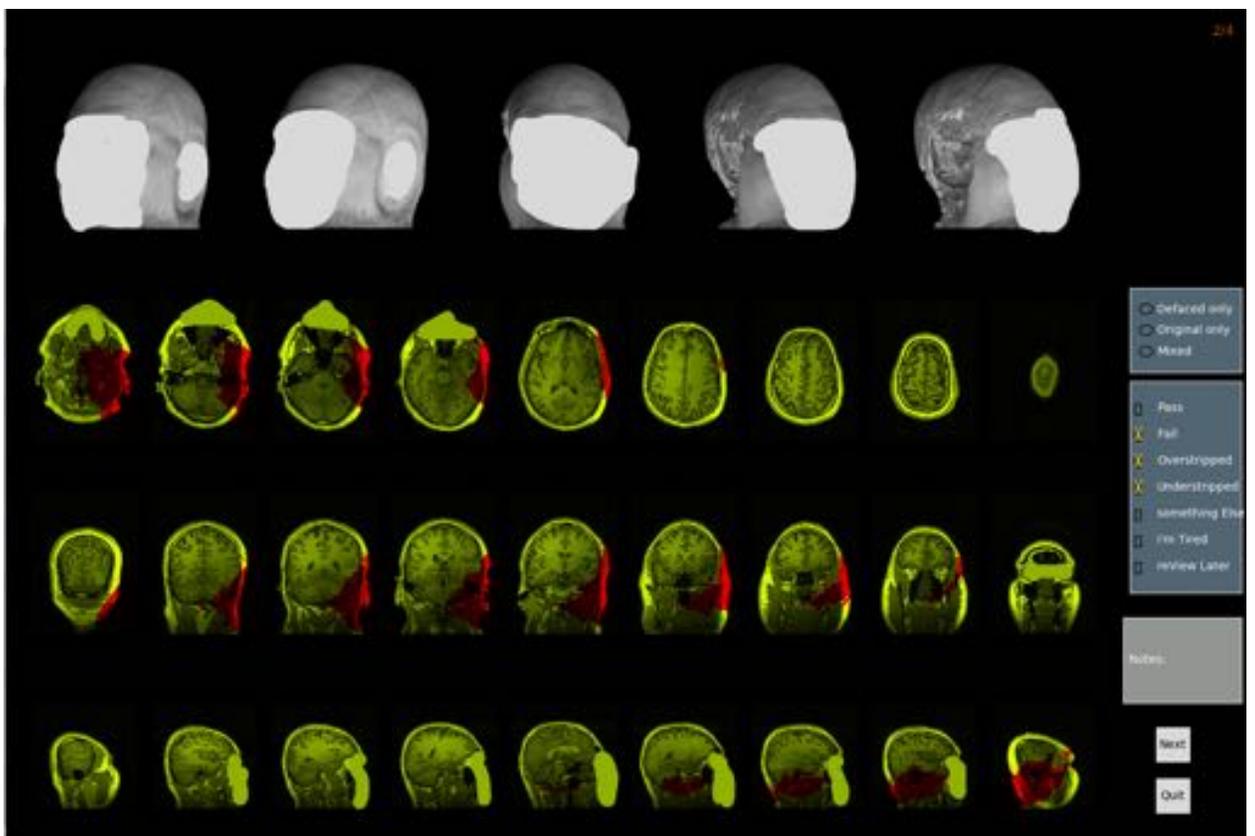
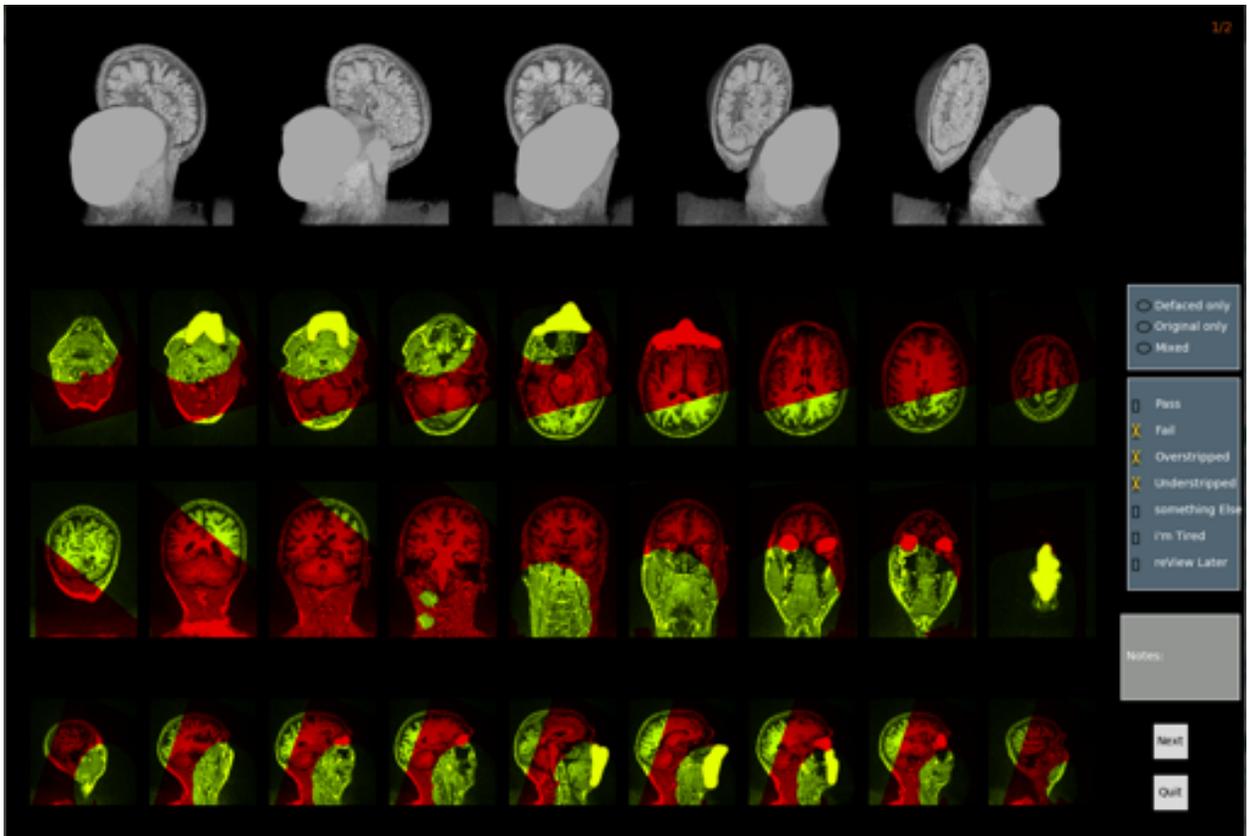


- 5.4.3 Starting defacing too far back from face surface: in some cases, the defacer confuses the location of the surface of the face and starts cutting from farther back than it should. This leaves most of the facial features intact and often ends up removing some of the brain tissue as well. This again should be marked as 'Fail', and 'Understripped'. 'Overstripped' should also be marked if some brain tissue is missing

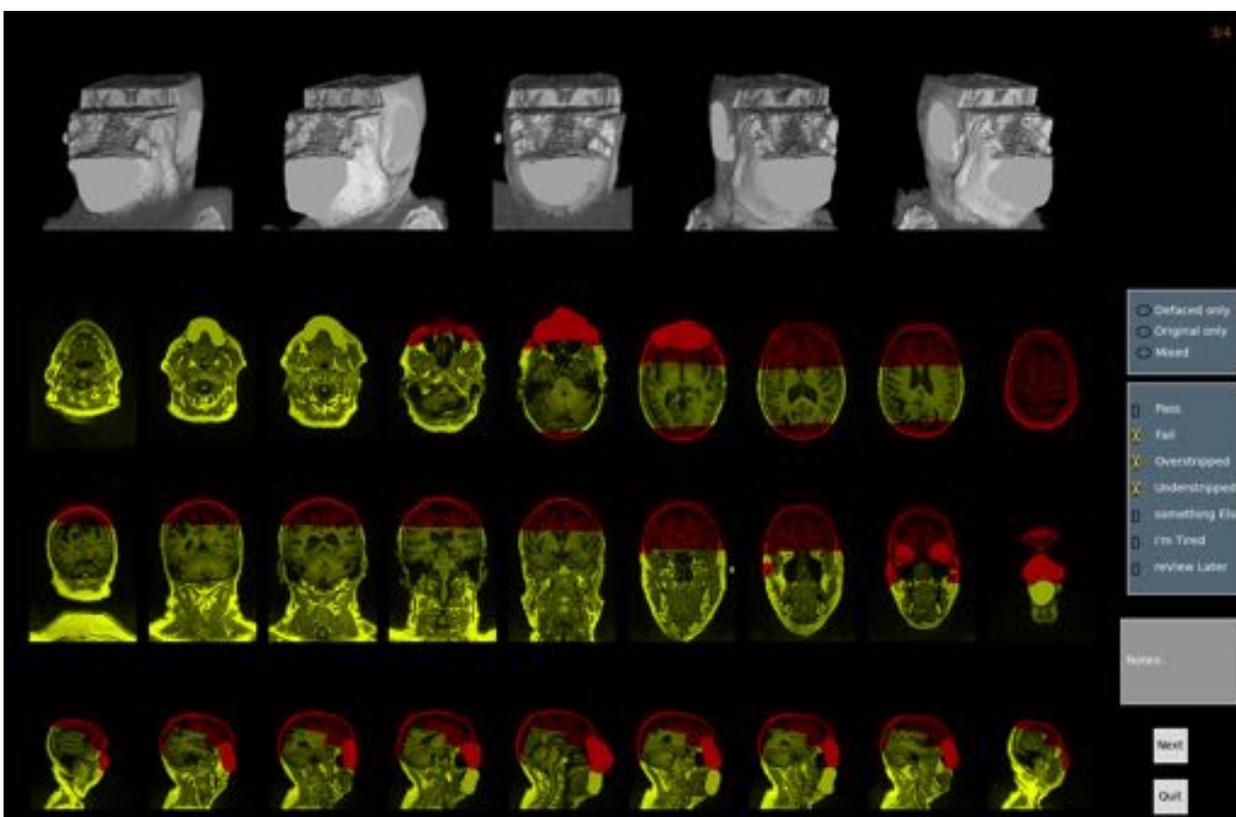
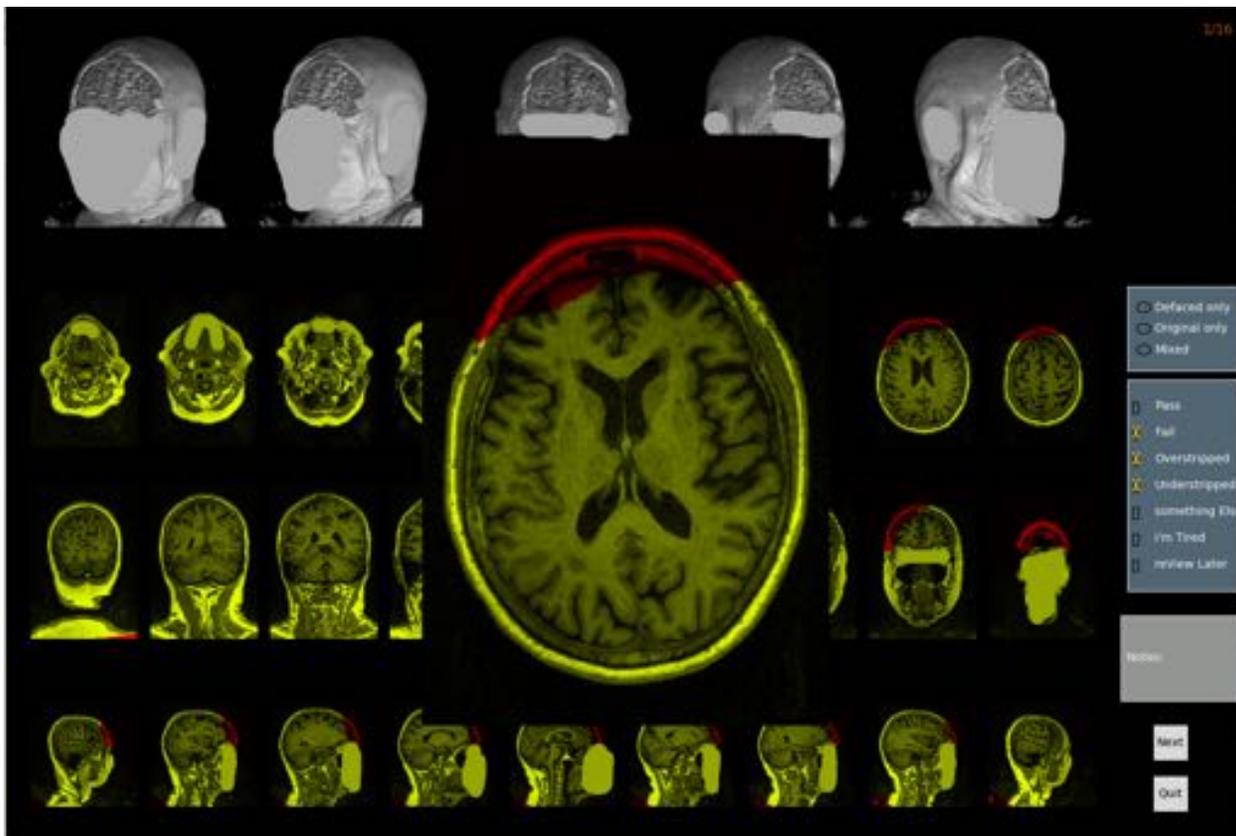


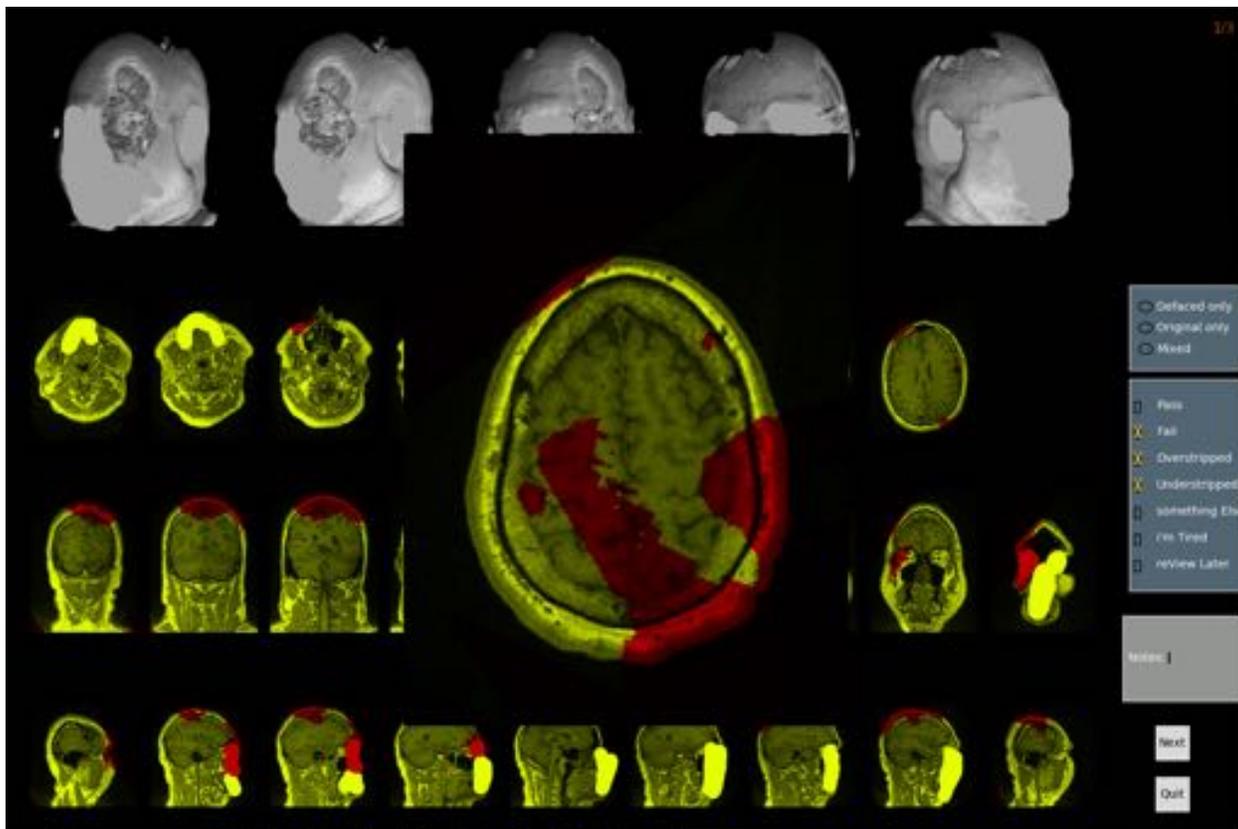
5.4.4 Mask misalignment: sometimes the alignment between mask that the defacer uses to remove identifiable features and the scan to be defaced, can fail, resulting in both in large sections of the brain being removed, and a wide portion of the face remaining. These cases should be rated as 'Fail', 'Overstrip' and 'Understrip'. Depending on how the alignment fails, this can present itself in multiple ways. Some examples are shown below.

VisualQC User Manual

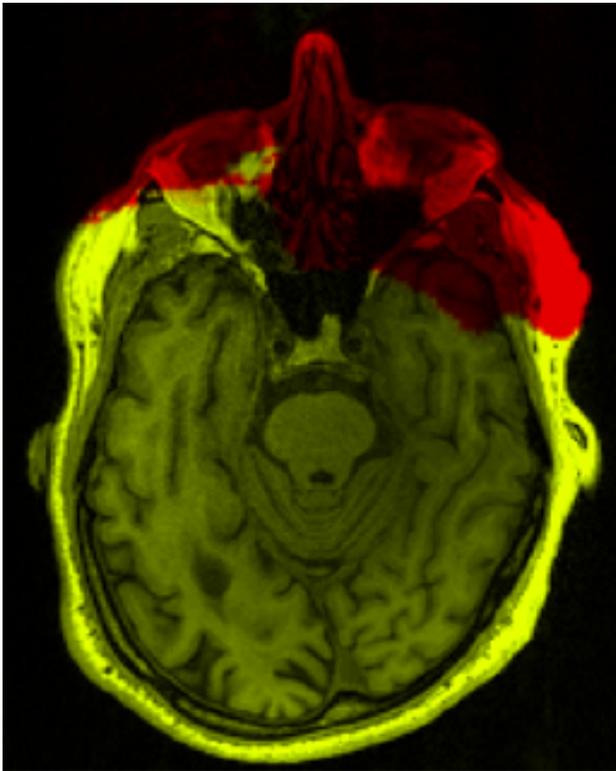


VisualQC User Manual

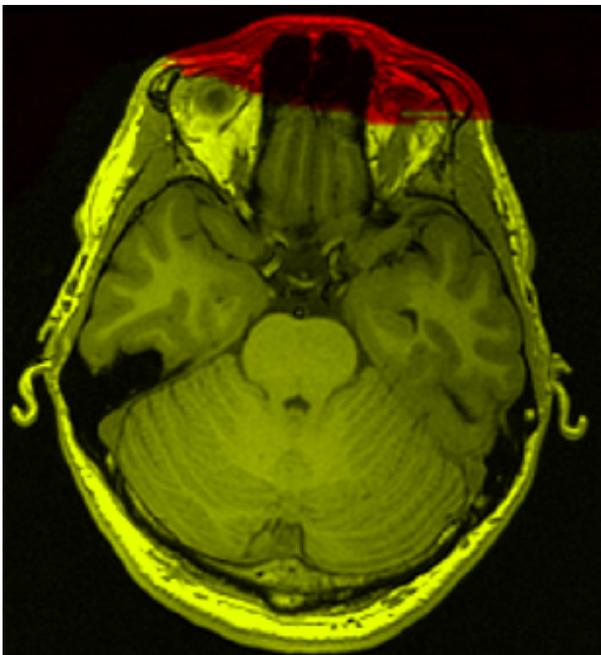




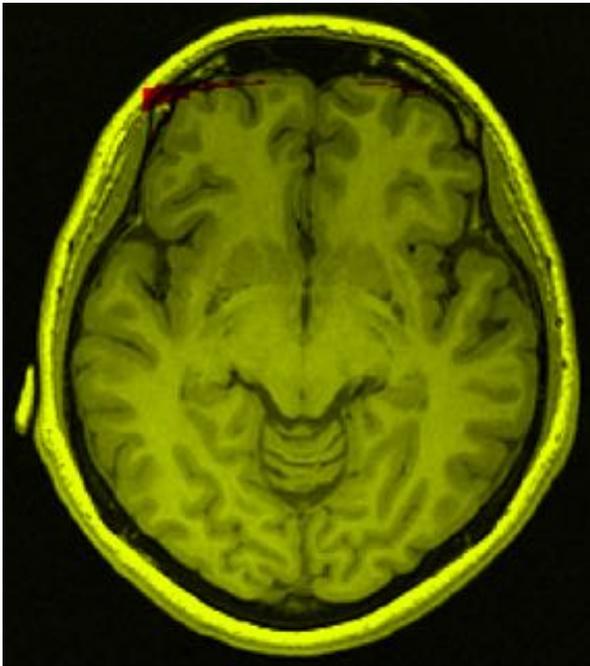
5.4.5 Overstripping: In some cases, the face will be removed perfectly, but the defacer is too aggressive and removes some of the brain as well. This is the error that you will have to watch most closely for. While sometimes the overstripping is obvious, such as below:



The defacer might only clip the edge of the brain:



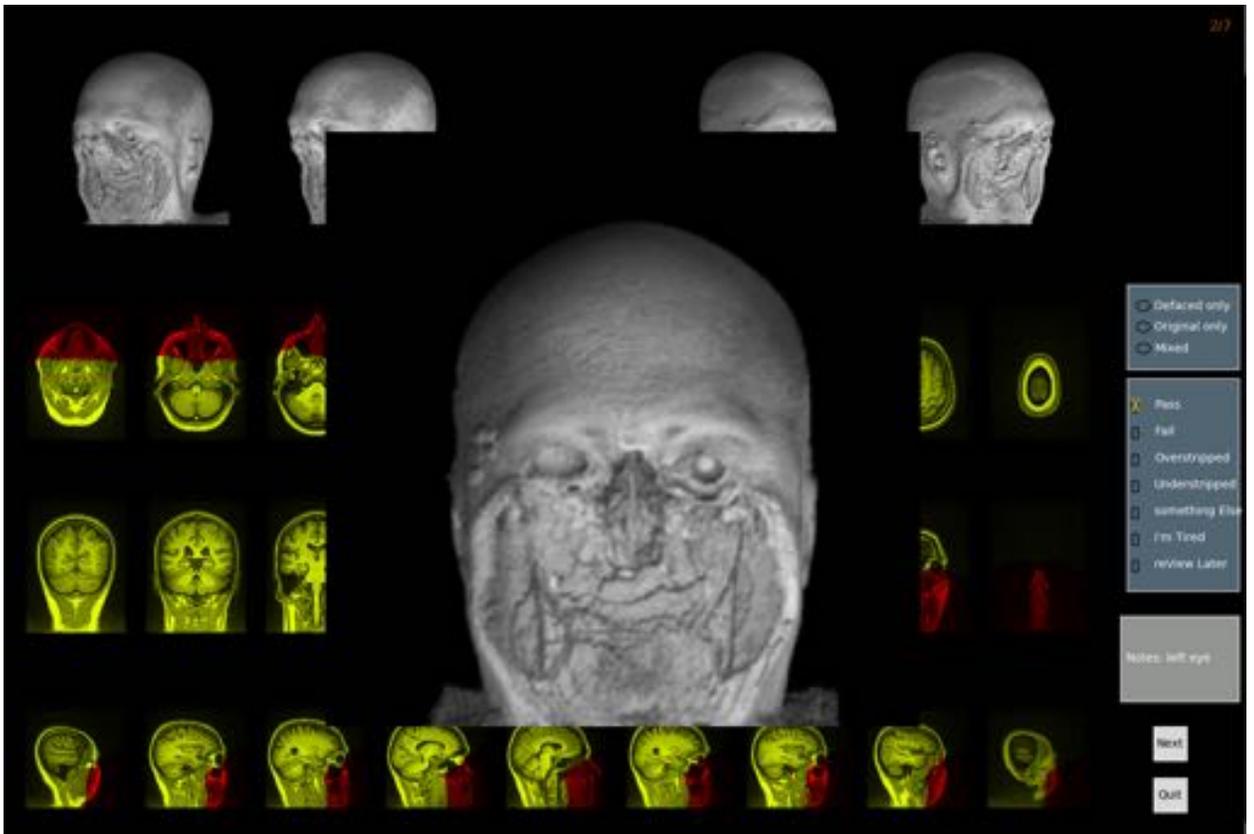
Or even cut into the brain away from the edge, depending on how the defacer determines what is face, and what is not:



In all cases, the scan should be rated as 'Fail' and 'Overstrip'.

- 5.4.6 Slight understrip: in an ideal case, the defacer strips off all of the face and does not touch any of the brain. But sometimes, the defacer will leave a small section of the face intact, such as a single eye or part of the mouth. If the brain is intact and this section is small enough that it could not be used to identify the person within the scan, then this can be marked as a 'pass', although a note should be added to indicate that the face was not completely stripped. If multiple defacers are being considered, this could be used to determine which is the best of the successful defacers

VisualQC User Manual

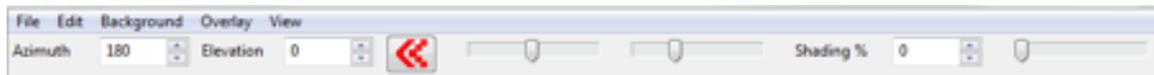


APPENDIX C: MRI 3D Render

There are a few different methods of creating a 3D rendered version of an MRI, in order to determine whether or not it still contains a face. Described here are two methods, one that is more manual, while the other can be automated. Either method, or another of your choice can be used for vqcdeface, so long as you are able to produce the required .png files of the renders.

5.5 Manual: mricron

Mricron is a NIfTI file viewer capable of producing 3D renders. (Available at: <https://people.cas.sc.edu/rorden/mricron/install.html>). Once a file has been opened, the render is produced by hitting Ctrl+R or by clicking Window>Render. A new window will open, and you will be able to adjust the angle of the render using Azimuth (0°-360°) and Elevation (-180°- 180°), brightness and contrast by sliding the two middle scale bars, and the shading (0-100%) for better visualization. To adjust for noise, you can adjust the air/skin threshold and search depth under Background. In most cases, a search depth of 4-8 voxels is good, while depending on the amount of noise the air/skin threshold should generally be set between 0% (no or very little noise) to 40% (noisy image). For extremely noisy images you may need to set this higher, but be forewarned, you may lose some of your actual image in doing this.



Once you are satisfied with the position and contrast of the render, you can then hit the  button to create a high resolution image, then hit Ctrl+S or File>Save as bitmap..., and select PNG under Save as type, then hit Save.

5.6 Automated: MATLAB or Octave

As long as your scans are not too noisy, or have similar levels of noise, you can automate the process using MATLAB or Octave. The following packages are required from the MATLAB file exchange: Viewer3D https://www.mathworks.com/matlabcentral/fileexchange/21993-viewer3d?s_tid=srchtitle and Tools for NIfTI and ANALYZE <https://www.mathworks.com/matlabcentral/fileexchange/8797-tools-for-nifti-and-analyze-image>. If using Octave or a version of MATLAB earlier than R2017a, you will also need: imrotate3_fast https://www.mathworks.com/matlabcentral/fileexchange/64002-imrotate3_fast, and imageresizen <https://www.mathworks.com/matlabcentral/fileexchange/64516-imresizen-resize-an-n-dimensional-array>. If using Octave, the image package must also be installed <https://octave.sourceforge.io/image/index.html>.

If using MATLAB, you can load the MRI scan using:

```
nii=load_nii('file.nii.gz');  
img=double(nii.img);
```

Since Octave can't handle .nii.gz files, you will need to add:

```
system('gunzip file.nii.gz'); then run nii=load_nii('file.nii');
```

After that, you will need to remove any outlier voxels with high signal intensity to prevent any scaling issues. Something like this is usually good:

```
img(img>(mean(img(img~=0))+3*std(img(img~=0))))=0;
```

Once this is complete, the image will need to be rescaled, first so that each voxel is 1mmx1mmx1mm, to remove distortion from scans with non-isometric voxels, then expanded so that the widest dimension is equal to 400, to maximize the amount of the produced render is taken up by the scan itself. If your images have a large FOV around the participant's head, you may wish to crop the image before resizing, to reduce the amount of whitespace.

```
if std(nii.hdr.dime.pixdim(2:4))~=0 || nii.hdr.dime.pixdim(2)~=1  
    pixdim=nii.hdr.dime.pixdim(2:4);  
    A=imresizen(A,[pixdim(1),pixdim(2),pixdim(3)]);  
end  
if max(size(A))<400  
    z=400/max(size(A));  
    A=imresizen(A,z);  
end
```

Next, you will need to set your render variables:

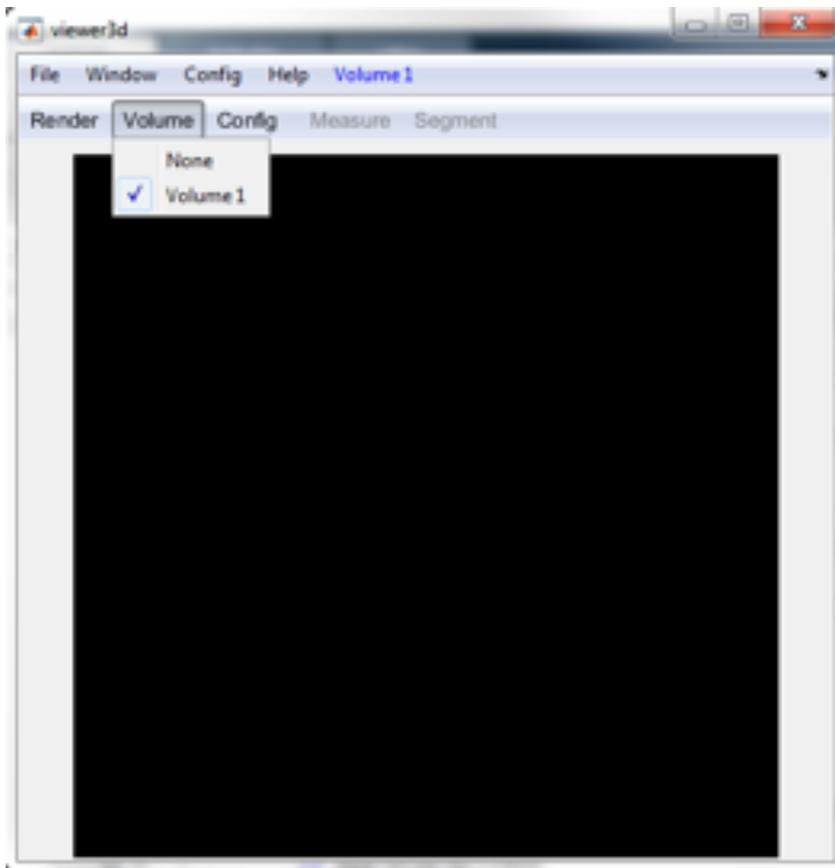
```
v1.RenderType='bw';  
v2.RenderType='shaded';  
[v1.AlphaTable,v2.AlphaTable]=deal([0 0 0 0.75 1 1 1 1 1 1]);  
v2.ColorTable=ones(3)/2;  
v2.ShadingMaterial='dull';
```

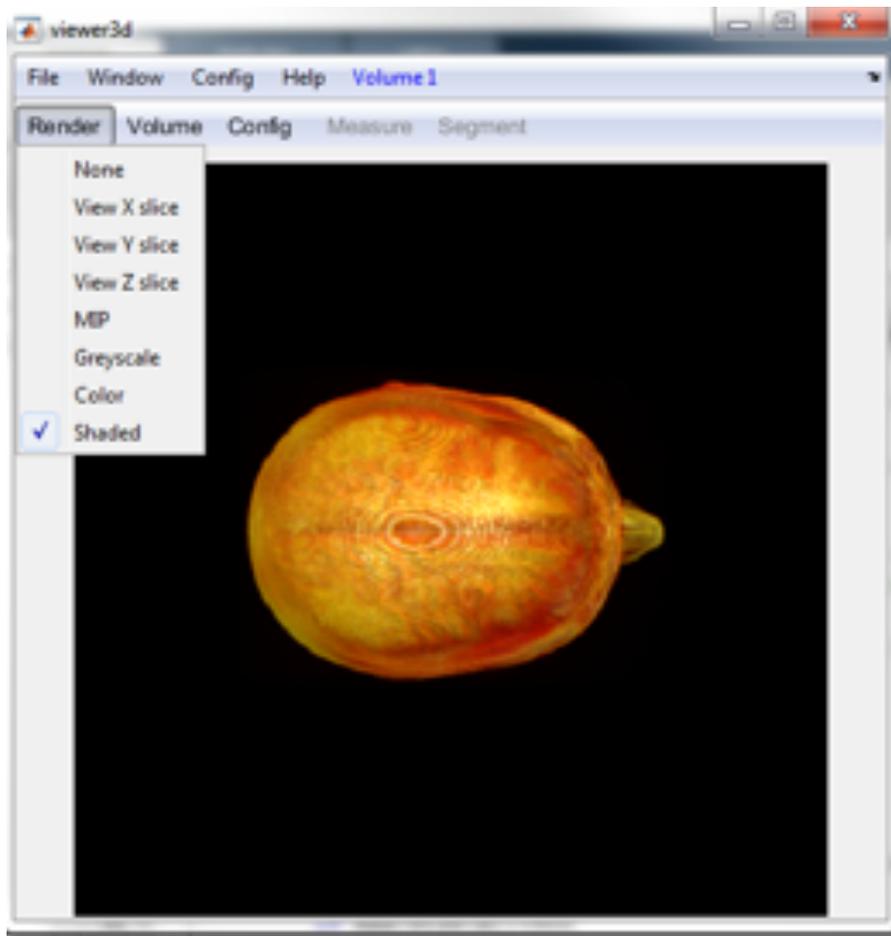
Most of these should remain the same, however, the AlphaTable may need to be changed, depending on how noisy your scans are. Each number represents the alpha value given to each voxel intensity represented in the table (e.g. in the code above, the AlphaTable has 10 values, the first 3 are 0s, the fourth is 0.75 and the rest are 1s. This means that the voxels with intensities of 0-30% of the maximum intensity are not included in the render, those in the 30-40% bracket are displayed at 75% opacity and anything higher will be completely opaque) Setting more of the levels to 0 will reduce the amount of noise, but be cautious

VisualQC User Manual

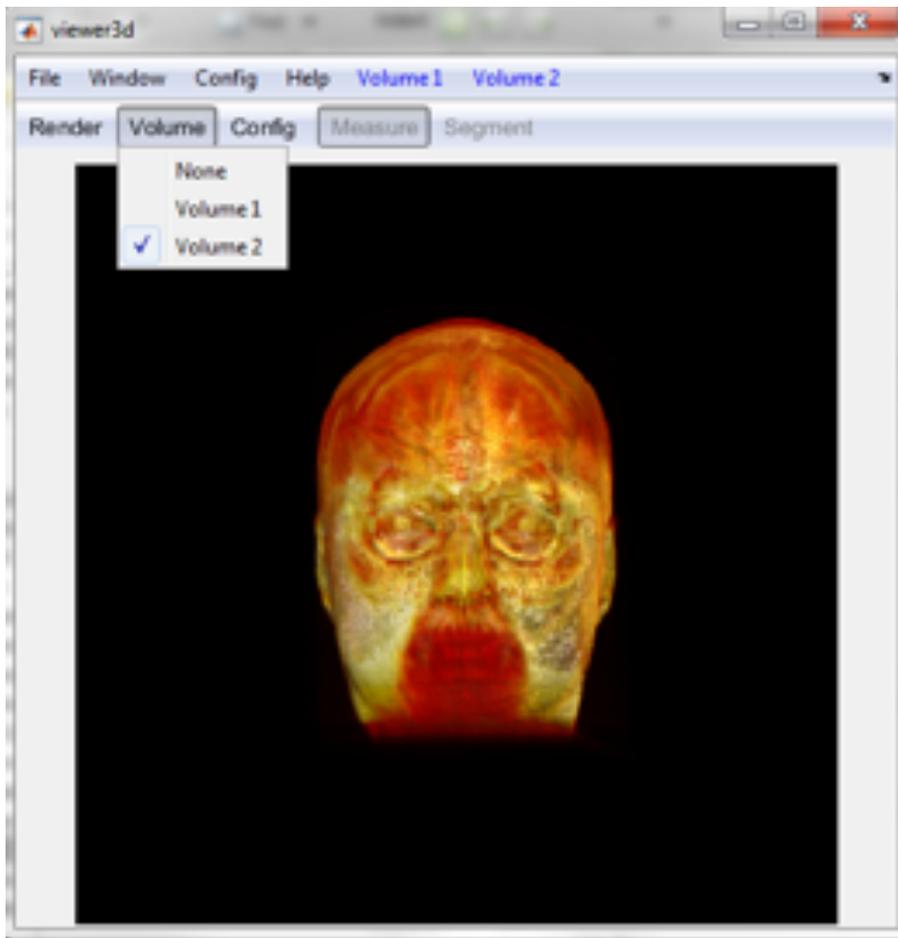
not to set them too low, or else you will end up masking some of the actual signal, this can even make it look as though the defacer has removed more of the face than it actually has. You may need to play around with these values until you find the ideal setting.

Depending on how your scans are oriented, you may need to rotate them so the render will be completed with the scan will be facing forward. This can be done by running the render once, opening the resulting image using `imshow`, then adjusting using `imrotate3_fast`. Alternatively, you can open the volume in `viewer3d`, using the command `viewer3d(A)`; A window will pop up like below. At first it will be blank; you will need to select `Volume>Volume 1`, then `Render>Shaded`, before you see anything.





As we can see in this example, the face is pointing to the right, instead of out of our screen, like we want. In this case, we can fix this by rotating the volume 90° around the x-axis, 90° around the y-axis and 180° around the z-axis, or `A=imrotate3_fast(A,[90 90 180]);` To double check that we've made the correct selection, we can reload the volume using `File>Load Workspace Variable`, selecting A, then selecting Load. To switch to the new volume, select `Volume>Volume 2`



Once you're satisfied with the orientation of the volume, you can make the render and save as a png using the following commands:

```
bw=render(A,v1);  
shade=render(A,v2);  
comp=bw.*shade./max(bw.*shade);  
comp=imgaussfilt(comp,1);  
imwrite(comp,'renderfile.png');
```

This will provide you with a straight view only; to make additional views, use `imrotate3_fast` to rotate the volume and rerun the previous code to render this new volume and save as a different png.