pyfMRIqc: a software package for raw fMRI data quality assurance

Journal of
open research software

Williams, B and Lindner, M 2020 pyfMRIqc: A Software Package for Raw fMRI Data Quality Assurance. *Journal of Open Research Software,* 8: 23. DOI: https://doi.org/10.5334/jors.280

SOFTWARE METAPAPER

pyfMRIqc: A Software Package for Raw fMRI Data Quality Assurance

Brendan Williams and Michael Lindner

Centre for Integrative Neuroscience and Neurodynamics, School of Psychology and Clinical Language Sciences, University of Reading, GB

Corresponding author: Michael Lindner (mlindner.mail@gmx.de)

pyfMRIqc is a tool for checking the quality of raw functional magnetic resonance imaging (fMRI) data. pyfMRIqc produces a range of output files which can be used to identify fMRI data quality issues such as artefacts, motion, signal loss etc. This tool creates a number of 3D and 4D NIFTI files that can be used for in depth quality assurance. Additionally, 2D images are created for each NIFTI file for a quick overview. These images and other information (e.g. about signal-to-noise ratio, scan parameters, etc.) are combined in a user-friendly HTML output file. pyfMRIqc is written entirely in Python and is available under a GNU GPL3 license on GitHub (https://drmichaellindner.github.io/pyfMRIqc/). pyfMRIqc can be used from the command line and therefore can be included as part of a processing pipeline or used to quality-check a series of datasets using batch scripting. The quality assurance of a single dataset can also be performed via dialog boxes.

Keywords: neuroscience; neuroimaging; fMRI; Python; quality control; artefacts
Funding statement: Brendan Williams is funded by the Magdalen Vernon PhD Studentship of the School of Psychology and Clinical Language Sciences, University of Reading. The project was funded by the Centre for Integrative Neuroscience and Neurodynamics, and the University of Reading.

(1) Overview

Introduction

Functional magnetic resonance imaging (fMRI) is a widely used research technique in human neuroscience to investigate experimentally driven changes in blood oxygenation (BOLD contrast) as a proxy of neural activity. Because fMRI data has a low signal-to-noise ratio, many different types of data quality issues can occur during data acquisition and these issues are related to a wide range of sources [1].

Sources that affect data quality during fMRI data acquisition include:

- Participant related artefacts.
  The behavior of the participant in the scanner has an influence on the data quality. For example, participant head motion is a primary source of motion artefacts during data acquisition as head motion interferes with the continuous measurement of the same tissue within a voxel. Participant head motion can also cause specific artefacts in fMRI data, such as spin-history artefacts [2] and motion-by-susceptibility interactions [3]. Another source of participant related artefacts are speaking and swallowing which can cause motion artefacts and localized changes in the homogeneity of the magnetic field creating further data quality issues [4]. Additional sources of participant related artefacts include physiological processes such as respiration, pulse and metabolism [5].
- Technique-related artefacts.
  A number of technique related artefacts can occur during data acquisition such as Gibbs artefacts, aliasing, zipper artefacts, nyquist, ghosting and many more (see a detailed overview: http://mriquestions.com/technique-related-artifacts.html).
- Tissue specific artefacts.
  Different molecules and tissues have different magnetic susceptibilities (diamagnetic, paramagnetic, superparamagnetic or ferromagnetic) and different combinations of these types of molecules and tissues could lead to various susceptibility artefacts such as distortions or local signal changes due to localized magnetic field in-homogeneities [6].
- Sequence specific artefacts.
  Many different types of imaging sequences are available for fMRI data acquisition, including echoplanar, multiband/multislice, and multi echo sequences. Each sequence type has its own specific advantages in terms of spatial or temporal resolution, but each also has its own sources and types of artefacts and other data quality issues, such as GRAPPA (GeneRalized Autocalibrating Partial Parallel Acquisition) artefacts in parallel imaging, or slice leakage artefacts in multiband acquisition [7, 8].

All these issues described above have different influences on fMRI data quality including: distorted images due to sequence specific artefacts; local incorrect values due to motion or changes in field homogeneity; change or loss of local or global signal. Some of these issues, such as motion, can be corrected if they are identified. Other issues, such as signal loss, cannot be corrected, and the data should be excluded from a research study to prevent spurious results from data analysis.

Human neuroimaging studies using fMRI involve the collection of large-scale datasets of BOLD contrast over time for each participant in the experiment. These datasets are then modelled to identify task relevant activity in the dataset. However, as previously mentioned, there are many potential sources of artefacts in fMRI data. These artefacts need to be identified to prevent the misidentification of artefacts as task-relevant activity during analysis. But, the manual screening of fMRI datasets for artefacts produced by the sources described above is a time-consuming process and can be error prone.

Current tools for quality assurance of fMRI data include VisualQC [9], and MRIQC [10]. Both provide the user with global measures of mean, standard deviation, and rate of signal change for functional timeseries. MRIQC also reports additional global metrics relating to spatial and temporal information, and artefact detection. pyfMRIqc extends the functionality of current tools by providing further quality assurance measures. These include slice-wise difference measures for consecutive functional volumes, nitfi files of quality assurance measures, and output viewable as a Html report. Like pyfMRIqc, both VisualQC and MRIQC are written in Python and can be initialized from the command line. However, the recommended method for using MRIQC involves running a containerized version in docker because the nipype workflow of MRIQC is dependent on the presence of additional analysis software. This is problematic since docker requires root access, raising potential security and confidentiality issues for neuroscience institutes who work with sensitive, personally identifiable data. pyfMRIqc, therefore, is more secure with respect to data protection as it operates independently from other fMRI analysis software packages, meaning it does not need to be containerized to work optimally.

pyfMRIqc has been developed as a piece of user-friendly software that is easy to install and use, providing the user with measures and guidelines for checking the quality of fMRI data. pyfMRIqc is an open-source software tool designed for quick and easy quality assurance of raw fMRI data before pre-processing or analysis via input dialog boxes (ID) or the command line. The software aims to help the user make decisions on the usability of their data, and any necessary pre-processing treatment.

Implementation and architecture

pyfMRIqc was written entirely in Python because it is free, widely used in neuroscience, and available for every major operating system (Linux, Windows and MacOS). pyfMRIqc has been developed and tested with Python version 3.6.4. pyfMRIqc can be executed using batch scripting if you wish to use the tool to check multiple datasets or include the tool as part of a pipeline. It can also be run using ID if you wish to check a single dataset, however not all input options are available if you chose to run pyfMRIqc via ID (more details below). pyfMRIqc is designed for the quality assurance of raw fMRI data, however pre-processed data can also be checked using the software.

Usage

pyfMRIqc can be used with ID (for beginners or for testing a single data set) or with command line parameters (recommended usage – for advanced users, multiple datasets, or adding to a pipeline). The ID version of pyfMRIqc needs to be started without any input parameters by typing the following into the terminal (note, the absolute path should be included if pyfMRIqc.py is not in your working directory):

```
python pyfMRIqc.py
```

For command line input the following parameter can be specified. (Some of these input parameters are exclusive for the command line parameter input and are not available via ID):

-n: functional MR nifti file (path).
-s: percentage of voxels with the lowest mean time course values (outside the mask) that are used for the SNR calculation (scalar value).

either

-t: threshold of minimum mean voxel intensity that should be included in the quality assurance (scalar value).

or

-k: binary nifti mask file of voxels that should be included in the quality assurance (path).

optional

-o: output directory (path).
-m: motion parameters file following motion correction with FSL (*.par), SPM (rp*.txt) or AFNI (*.1D) (path).
-x: if -x is set, the 3D and 4D nifti output files are not saved.
-h, --help: prints the pyfMRIqc input help to the command window.

For example:

```
python pyfMRIqc.py \
     -n <your_functional_file.nii> \
     -s 10 \
     -k <your_mask_file.nii> \
     -o <your_output_path> \
     -m <your_motion_file> \
     -x
```

Would execute pyfMRIqc for your_functional_file.nii. The mask your_mask_file.nii would be applied to your_functional_file.nii, 10% of voxels (with the lowest mean time course values) would be used for SNR calculation and

your_motion_file would be loaded. pyfMRIqc output data would be saved to your_output_path and additional nifti files would not be saved as part of your output.

Measures and output

Mean

The mean voxel intensity over time can be used to see if there is any general signal loss. pyfMRIqc provides 3D nifti and a 2D .png overview image as output for the mean voxel intensity.

Variance

The variance over time is calculated voxel-wise. Typically, grey matter, brain stem, eyes, and blood vessels have the highest variability of BOLD signal. Therefore, these tissue types have higher values and are brighter in this image (nifti and 2D). Sudden or unexpected signal changes due to artefacts, motion etc. in one or more volumes will increase the variance and therefore will be detectable in these images. For example, **Figure 1** shows how local signal dropout in a few volumes will appear as a grey pattern in the image. pyfMRIqc provides 3D nifti and a 2D .png overview image as output for the variance.

Masks

pyfMRIqc creates two output .nii files containing binary masks:

1) Mask: depending on the user input (-t or -k):
   a. containing voxels with higher intensity values than given threshold (t) (**Figure 2**, blue)
   or
   b. same mask as the input mask (k)
2) Mask for Signal to Noise Ratio (SNR): contains the voxels used in the SNR calculation depending on input -s (**Figure 2**, green)

Mean voxel time course of bins

To produce this output, voxels are sorted depending on their mean voxel time course intensity and then divided into 50 bins with an equal number of voxels in each bin. Then the voxels are averaged per volume for each bin. Each horizontal line represents the averaged voxel time course of one bin. The bins are sorted ascending (top-down) depending on the mean voxel intensity, so that the upper part of the plot represents voxels outside the brain, lower middle parts represent

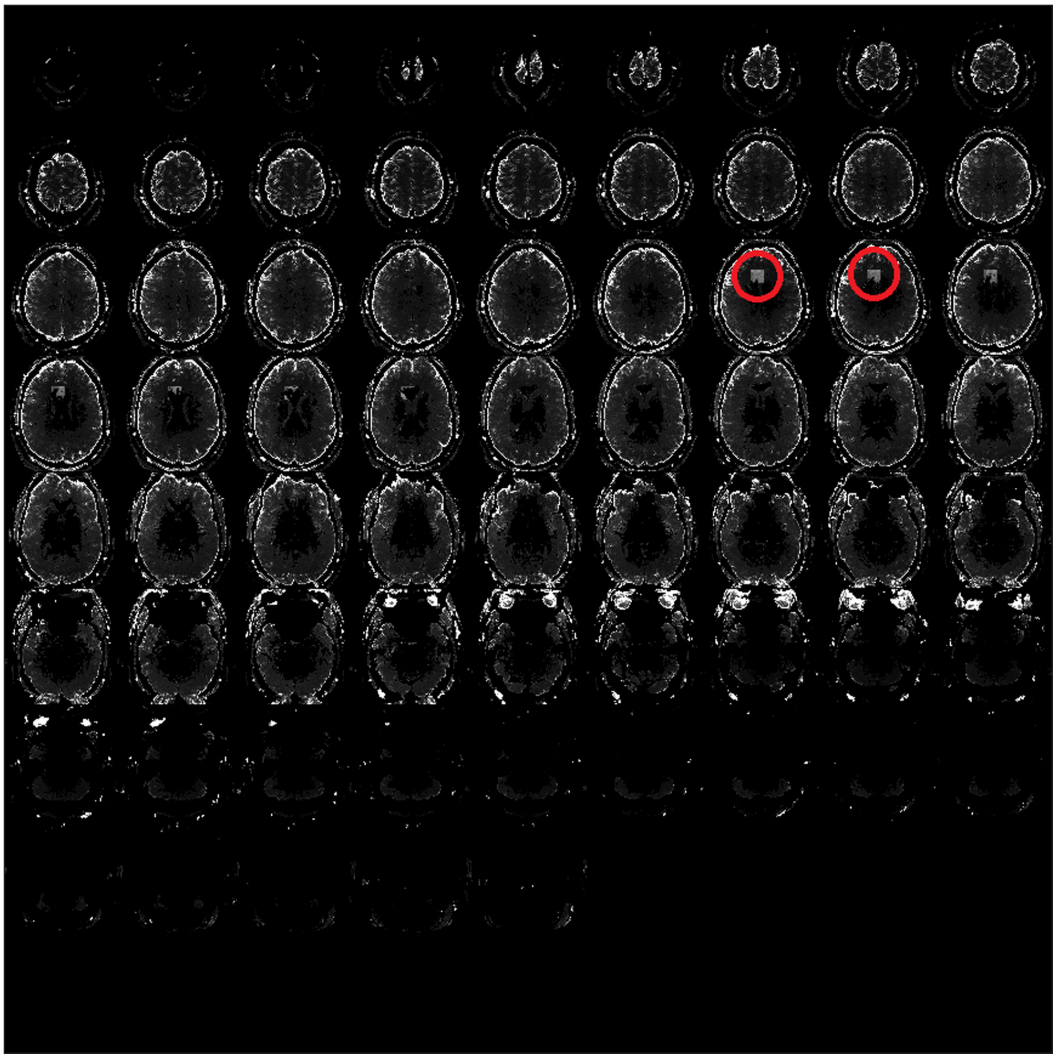

Figure 1: Example of cubic local signal dropout in the variance image. The grey squares show the position of local signal loss in the example file pyfMRIqc_example_local_signal_loss.nii.gz.

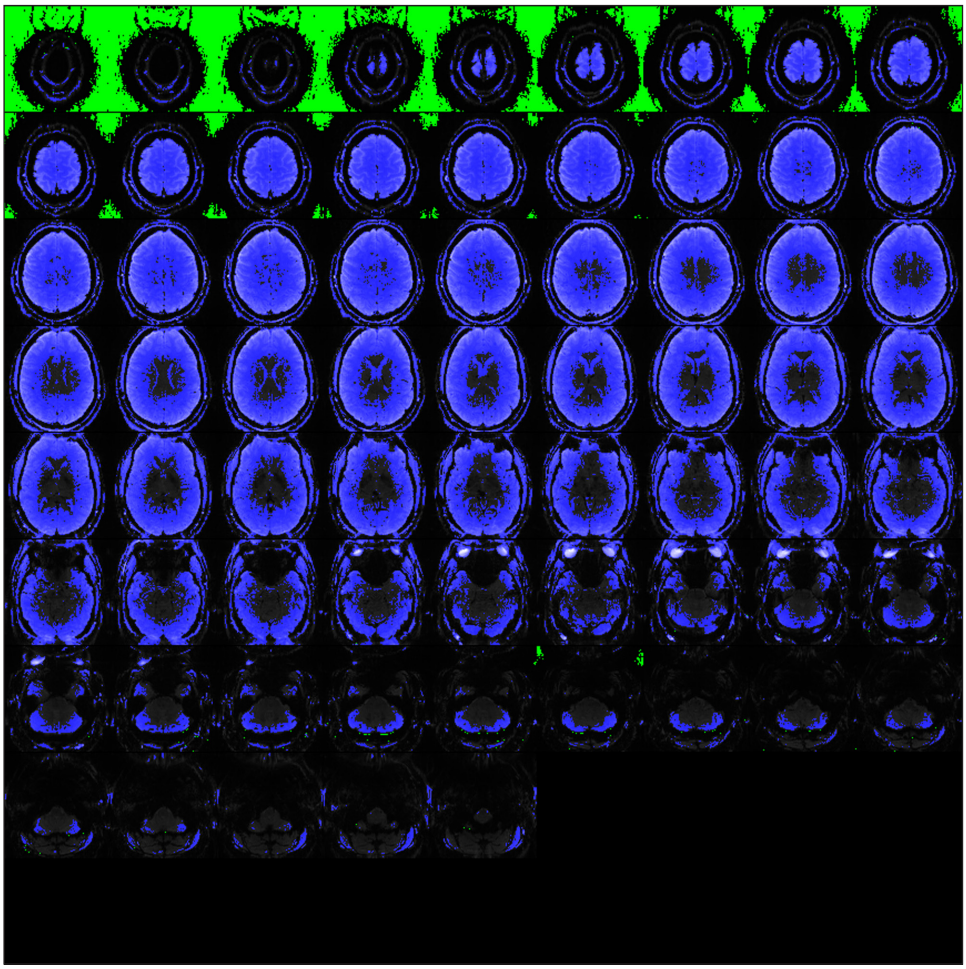

Figure 2: Mask image for file pyfMRIqc_example_local_signal_loss.nii.gz. The blue color shows voxels above the given intensity threshold that are included in the quality assurance. Green voxels are the n^{th} percentage of voxels with the lowest mean time course values used for SNR calculation.

grey matter, and lower parts represent voxels in white matter.

This image is meant to detect unexpected or unwanted changes in the signal variance as described by Power [11]. Prominent vertical patterns in this image could be related to different types of issues such as signal loss, noise and artefacts, which in turn may be caused by motion, respiratory interference, field heterogeneity issues, etc (see **Figure 3A**). It can also help to identify if an issue is global (from top to bottom, **Figure 3B**) or only related to some of the tissue types (vertical pattern in subparts). Additionally, the user can check if any known effect (such as motion – e.g. as identified from the motion plot, **Figure 4A**) influences the signal change or not. pyfMRIqc provides a 2D .png image as output for the mean voxel time course of bins.

Scaled squared difference

For the scaled squared difference (SSD), the first derivative of the time course `q = dx/dt, dt = 1` is calculated. This yields the difference of two consecutive time points `t - (t-1)` for each voxel (V). These differences are then squared to handle negative values and to improve scaling for the identification of outliers. To take the differences in absolute signal intensity into account the squared differences are divided by the mean of squared differences of all voxels (M):

$$SSD_t = \frac{(V_t - V_{t-1})^2}{M}$$

Four quality assurance images are created using SSD values. The first image shows the mean SSD over all voxels to get an overview of any global signal changes. The second image shows the mean SSD for each slice separately. Each colored plot in the image represents one slice. In contrast to the first image, signal changes in the second image can be detected slice-wise, giving an overview of non-global sudden signal changes. The third image shows the normalized average of the demeaned voxel intensity, the normalized SSD variance, and the normalized sum of relative movement for each volume. This plot can be used to directly compare the quality assurance report of multiple datasets since the values are normalized. The fourth image displays the minimum, mean and maximum slice-wise SSD averaged over volumes.

SSD can be used to detect sudden changes in signal intensity which can occur because of different problems, for example:

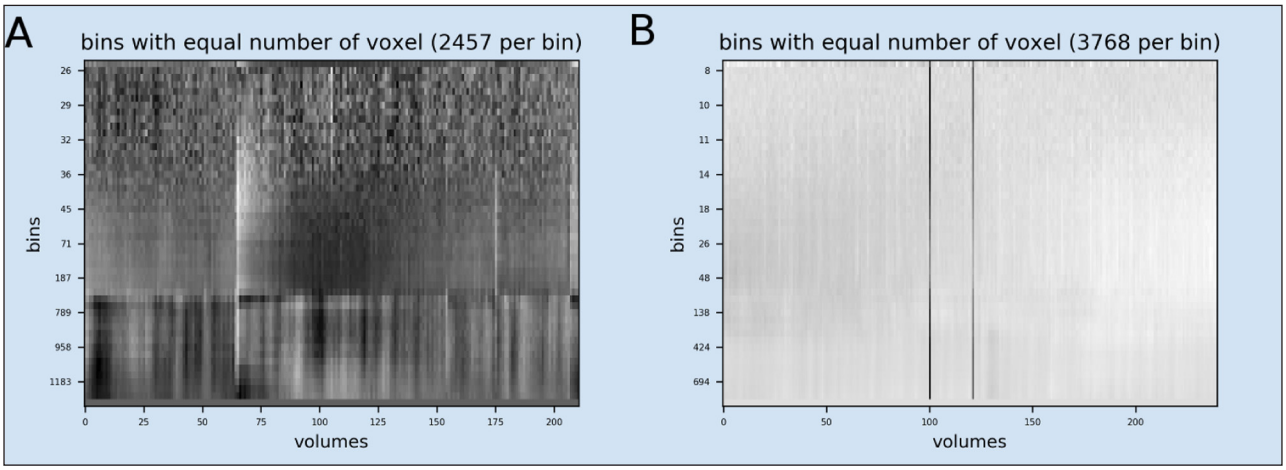

Figure 3: Examples of the mean voxel time course of bins plot **A)** Example file pyfMRIqc_example_motion.nii.gz: Two major vertical patterns show the global effects of head motion on different tissue types (bins 32 and higher). The first vertical pattern after volume 65 is related to a strong movement and the second pattern after volume 175 is related to another small movement. The lower numbered bins (top of the plot) represent voxels covering empty space around the head that are not affected by motion. **B)** Example file pyfMRIqc_example_volume_intensity_loss.nii.gz: Two major vertical patterns between volumes 100 and 122 show global signal intensity loss.

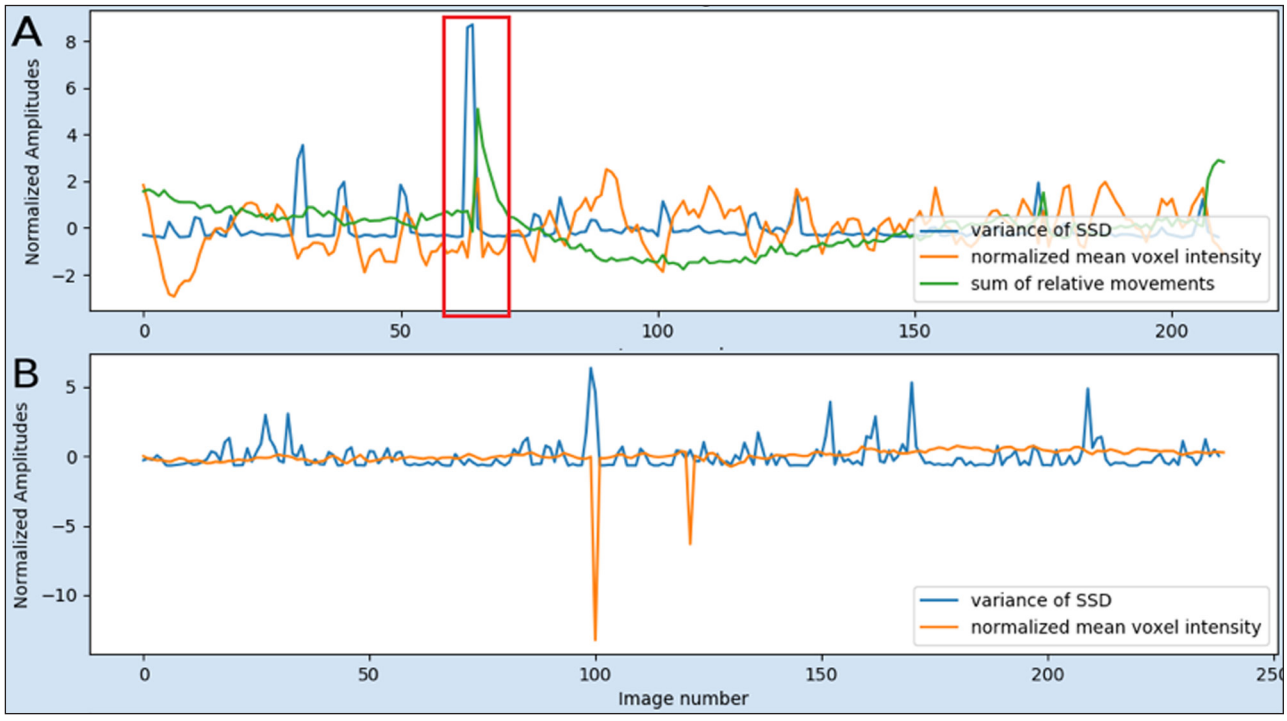

Figure 4: Example of the third scaled squared difference plot: **A)** In the red box the plot shows that the variance of the SSD (blue line) is related to the head motion of the participant (green line). This is plotted from example file pyfMRIqc_example_motion.nii.gz. **B)** The negative peaks in the mean voxel intensity are related to global signal intensity loss in two volumes in the example file pyfMRIqc_example_volume_intensity_loss.nii.gz.

- head motion (**Figure 4A**) – at the time point the motion occurs the voxel time course of a spatially defined brain tissue is interrupted. In this case the SSD will show either a peak or step (positive or negative).
- signal loss or artefact (**Figure 4B**) – the "continuous" voxel time course signal is interrupted by sudden extreme low or high values. In these instances, the image would show a large positive or negative peak.

pyfMRIqc provides 3D nifti and a 2D .png overview image as output for the SSD.

Signal-to-noise ratio

To calculate the signal-to-noise ratio, the mean voxel intensity for each voxel is divided by the standard deviation of the mean noise, averaged across the temporal domain. In pyfMRIqc the mean noise is defined as the

average of the s% of voxels having the lowest mean intensity, where s is dependent on the user input (-s). pyfMRIqc provides a 3D nifti image with voxel-wise SNR values and a summary of values as output for the SNR.

The Slice SNRs measures the time course SNR averaged across each slice, and Mean voxel SNR is the average over all the slices together. The higher the SNR, the smaller the relative fluctuations and more stable the signal is over repeated measurements. The SNR provides an estimate of the reliability (~ reproducibility) of fMRI data and serves as a general goodness measure.

Optional: Motion parameter summary

If a motion parameter file is provided as input, a short summary is provided at the end of the html file, namely the mean and max of each absolute and relative movement, where absolute movement is movement relative to the first volume, and relative movement is the movement relative to the previous volume.

Sudden movements are worse for fMRI data (and motion correction algorithms) than slow drifts. Therefore, relative movement values are the more important values. As a rule-of-thumb relative movement that is bigger than the acquisition voxel size is unacceptable and therefore, if pyfMRIqc shows a number bigger than 0 for "relative movement > voxel size", the data should not be used. Relative movement > 0.5mm is not good and thorough checks should ensure that the motion correction algorithm has worked properly. Relative movement > 0.1mm may be ok but in case of high numbers in this metric the data should be checked thoroughly and used with care. Additionally, ideally the max absolute motion should be less than 2mm or at least less than the voxel size.

HTML output file

pyfMRIqc provides an html file to combine quality assurance values and all the 2D images mentioned above for a quick overview of the data quality.

Output nifti files

nifti images (if -x is not set):

- mean_<yourfile>
  mean voxel intensity over time (3D)
- variance_<yourfile>
  variance of the voxel time courses(3D)
- mask_<yourfile>
  binary – containing voxels above the threshold or the input mask (3D)
- mask4snr_<yourfile>
  binary – lowest n percent of lowest values used for SNR calculation (3D)
- snr_<yourfile>
  voxel-wise signal-to-noise ratio (3D)
- squared_scale_diff_<yourfile>
  squared scaled signal variability: squared difference between two consecutive volumes divided by the global mean difference

Quality control

pyfMRIqc is currently used by members of the neuroimaging community at the University of Reading for quality assurance of raw fMRI data. We have created example datasets with specific deliberate errors to test and validate the algorithms in pyfMRIqc. These deliberate errors are representative of the type of errors that could occur during fMRI data acquisition. These example datasets are also freely available via GitHub for users to download and trial pyfMRIqc. The example file *pyfMRIqc_example_motion.nii.gz* is an example of participant motion. This data set contains 211 volumes of a standard EPI sequence. The participant moved twice during the scan, the first is a strong movement after volume 65 and the second is a small movement after volume 175. The example file *pyfMRIqc_example_local_signal_loss.nii.gz* is an example of local signal loss (signal loss has been artificially created for this example by replacing signal with low gaussian random noise). Signal loss of a cube of voxels (10 × 10 × 10) is located in the middle of the frontal cortex for two volumes (50 and 120). The example file *pyfMRIqc_fMRI_example_volume_intensity_loss.nii.gz* is an example of a global reduction in signal intensity for two volumes: Volume 101 shows an intensity loss of about 20% of the mean intensity and volume 122 an intensity loss of about 10% of the mean intensity.

Although pyfMRIqc has undergone extensive testing to ensure proper functionality, users are able to contact the developers with any issues or bugs they may discover when using pyfMRIqc using the projects dedicated email address: pyfMRIqc@gmail.com. Additionally, as pyfMRIqc is open-source (GPLv3) and freely available, we actively welcome contributors from the neuroscience community to contribute to the project if they wish. A detailed description of how to use pyfMRIqc can be found in the pyfMRIqc documentation on the GitHub page (https://drmichaellindner.github.io/pyfMRIqc/).

(2) Availability

Operating system

pyfMRIqc is written in Python and therefore is available on any operating system that supports Python frameworks. pyfMRIqc was tested on Ubuntu 18.04, Windows 7 and 10, and macOS Sierra.

Programming language

Python 3.6 (tested on Python 3.6.4)

Additional system requirements

None

Dependencies

The following Python packages need to be available, version number used for development is provided in brackets. Newer versions of packages should also be compatible:

nibabel (2.2.1)
numpy (1.14.3)
matplotlib (2.1.0)
scipy (0.19.1)
easygui (0.98.1)

List of contributors

Michael Lindner is the main developer and maintainer of pyfMRIqc. Brendan Williams has contributed to some of the functions and documentation. Both authors were involved in writing this article. The code for this software was developed at the Centre for Integrative Neuroscience and Neurodynamics (CINN) at the University of Reading, UK.

Software location

Code repository

Name: GitHub
Identifier: https://github.com/DrMichaelLindner/pyfMRIqc
Version: 1.1
License: GNU GPL3
Date published: 15/05/19

Language

English

(3) Reuse potential

pyfMRIqc is written in Python, a widely used and freely available programming language. Because the software is developed in Python it is nearly platform-independent and therefore can be used by the majority of individuals in the neuroimaging community. Because pyfMRIqc is a user-friendly tool it is suitable for neuroscientists at all stages of their career and can be used for quality assurance for both individual datasets or can be implemented into automatic pipelines for data quality assurance immediately after scanning. Indeed, the latter usage is similar to how pyfMRIqc is currently being implemented within the CINN imaging community, where data quality assurance using pyfMRIqc is now standard procedure. Future releases of pyfMRIqc may also include additional features that will increase the reusability of the tool. Support can be provided by the authors via email (pyfMRIqc@gmail.com).

Competing Interests

The authors have no competing interests to declare.

References

1. **Erasmus, L, Hurter, D, Naude, M, Kritzinger, H** and **Acho, S** 2004 A short overview of MRI artefacts. *South African Journal of Radiology*, 8(2): 13. DOI: https://doi.org/10.4102/sajr.v8i2.127
2. **Friston, K, Williams, S, Howard, R, Frackowiak, R** and **Turner, R** 1996 Movement-Related effects in fMRI time-series. *Magnetic Resonance in Medicine*, 35(3): 346–355. DOI: https://doi.org/10.1002/mrm.1910350312
3. **Wu, D, Lewin, J** and **Duerk, J** 1997 Inadequacy of motion correction algorithms in functional MRI: Role of susceptibility-induced artifacts. *Journal of Magnetic Resonance Imaging*, 7(2): 365–370. DOI: https://doi.org/10.1002/jmri.1880070219
4. **Birn, R, Bandettini, P, Cox, R, Jesmanowicz, A** and **Shaker, R** 1998 Magnetic field changes in the human brain due to swallowing or speaking. *Magnetic Resonance in Medicine*, 40(1): 55–60. DOI: https://doi.org/10.1002/mrm.1910400108
5. **Krüger, G** and **Glover, G** 2001 Physiological noise in oxygenation-sensitive magnetic resonance imaging. *Magnetic Resonance in Medicine*, 46(4): 631–637. DOI: https://doi.org/10.1002/mrm.1240
6. **Hood, M, Ho, V, Smirniotopoulos, J** and **Szumowski, J** 1999 Chemical Shift: The Artifact and Clinical Tool Revisited. *RadioGraphics*, 19(2): 357–371. DOI: https://doi.org/10.1148/radiographics.19.2.g99mr07357
7. **Todd, N, Moeller, S, Auerbach, E, Yacoub, E, Flandin, G** and **Weiskopf, N** 2016 Evaluation of 2D multiband EPI imaging for high-resolution, whole-brain, task-based fMRI studies at 3T: Sensitivity and slice leakage artifacts. *NeuroImage*, 124: 32–42. DOI: https://doi.org/10.1016/j.neuroimage.2015.08.056
8. **McNabb, CB, Lindner, M, Shen, S, Burgess, LG, Murayama, K** and **Johnstone, T** 2020 Inter-slice leakage and intra-slice aliasing in simultaneous multi-slice echo-planar images. *Brain Struct Funct*. DOI: https://doi.org/10.1007/s00429-020-02053-2
9. **Raamana, P** 2018 VisualQC: Assistive tools for easy and rigorous quality control of neuroimaging data. DOI: https://doi.org/10.5281/zenodo.1211365
10. **Esteban, O, Birman, D, Schaer, M, Koyejo, O, Poldrack, R** and **Gorgolewski, K.** MRIQC: Advancing the automatic prediction of image quality in MRI from unseen sites. *PLoS ONE*, 12(9): e0184661. DOI: https://doi.org/10.1371/journal.pone.0184661
11. **Power, J** 2017 A simple but useful way to assess fMRI scan qualities. *NeuroImage*, 154: 150–158. DOI: https://doi.org/10.1016/j.neuroimage.2016.08.009

How to cite this article: Williams, B and Lindner, M 2020 pyfMRIqc: A Software Package for Raw fMRI Data Quality Assurance. *Journal of Open Research Software*, 8: 23. DOI: https://doi.org/10.5334/jors.280

Submitted: 05 June 2019 **Accepted:** 16 September 2020 **Published:** 07 October 2020

]u[*Journal of Open Research Software* is a peer-reviewed open access journal published by Ubiquity Press.

OPEN ACCESS