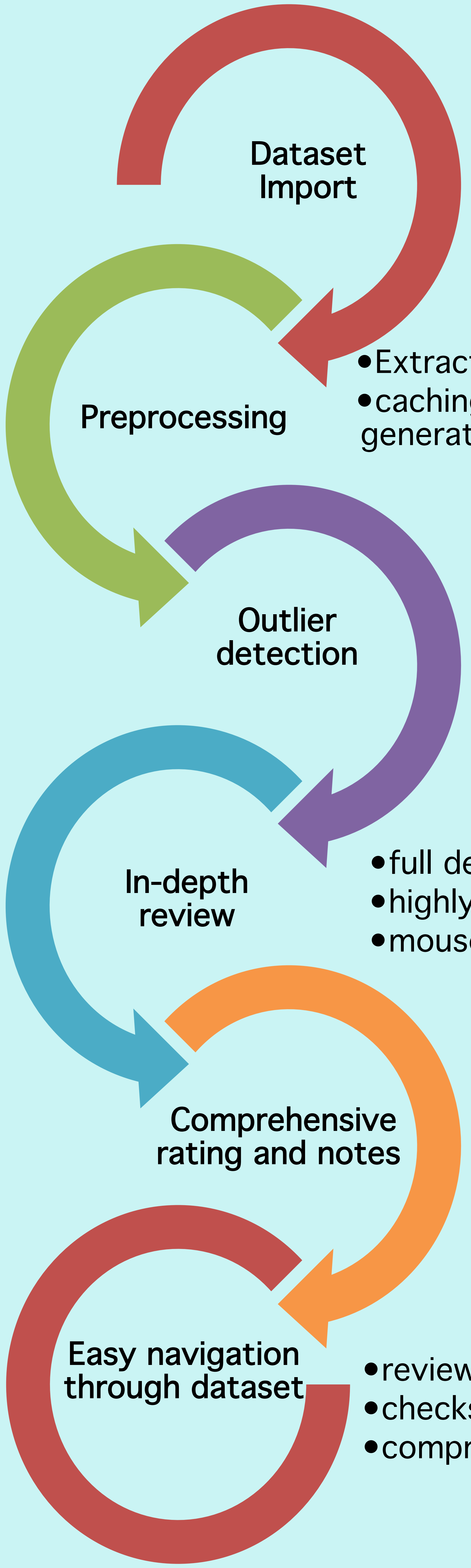
# VisualQC: rigorous yet easy quality control of neuroimaging and medical data

visit: github.com/raamana

## Introduction

- Assessing the quality of neuroimaging data: raw MR acquisition (functional or diffusion) or automatic segmentation (gray/white surfaces) requires human visual inspection
- Given the complex 3D/4D nature of image volumes, this requires inspection in all 3 planes and multiple cross-sections. Often, looking at raw data is not sufficient, especially to spot subtle errors, wherein statistical measurements can greatly assist in identifying the artefacts and rating their severity
- For certain cases (e.g. assessing the accuracy of cortical sheet or in reviewing an fMRI time series), multiple types of visualizations (surface-rendering or carpet plots) and metrics (e.g. SNR) need to be taken into account
- This process is cumbersome, time-consuming and prone to human errors e.g. paying insufficient attention to detail or inaccurate bookkeeping
- As the datasets have bigger sample sizes and more modalities, there is a great need for a fast tool for quality control (QC) of neuroimaging data
- QC has been studied in different dimensions, including developing image quality metrics, detecting unusable scans, and a mix of both. However, a common lesson learnt from from multiple modalities is that the accuracy of these "automatic" methods is too low to be relied on for routine usage and that manual visual inspection is necessary
- The VisualQC package, purpose-built for rigorous QC, aims to reduce this laborious process to an easy to use workflow
- The API is extensible and adding new use cases is as simple as writing two functions to customize data import and visualization.

## Workflow


Dataset Import
Preprocessing
Outlier detection
In-depth review
Comprehensive rating and notes
Easy navigation through dataset

Dataset Import:
- support for popular formats
- load in only what is needed
- regex retrieval of data possible

Preprocessing:
- Extract [image] quality metrics
- caching heavy processing e.g. pre-generate CPU-intensive visualizations

Outlier detection:
- Multivariate (any quality metric)
- high-dimensional, yet fast
- on the dataset as a whole, or per slice in a volume, or per time point or gradient

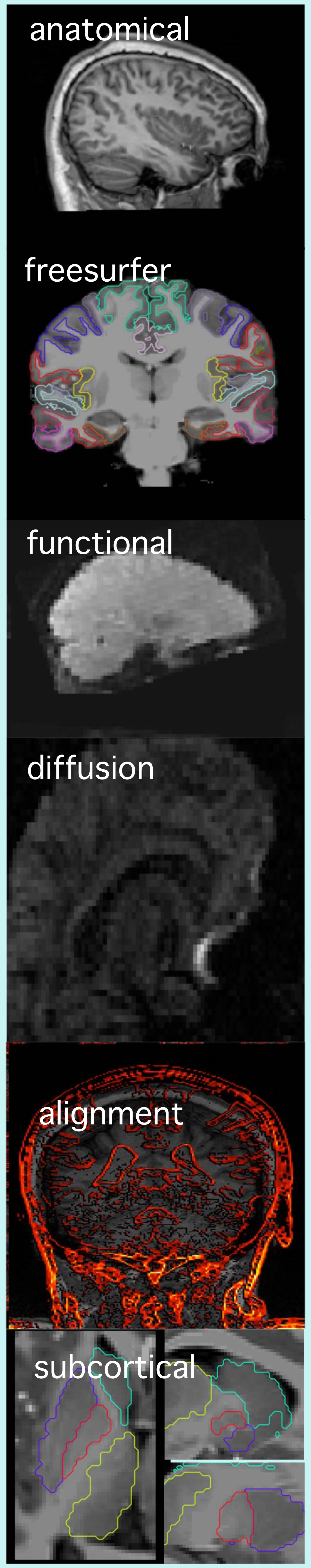
In-depth review:
- full details and alternative forms
- highly interactive and customizable
- mouse / keyboard shortcuts

Comprehensive rating and notes:
- enumerated list of artefacts
- simple checkbox/radio buttons
- easy notes taking in-place

Easy navigation through dataset:
- review next subject with a single click
- checks to ensure review/rating is done
- comprehensive record of review activity

open source + pure python

## Use cases



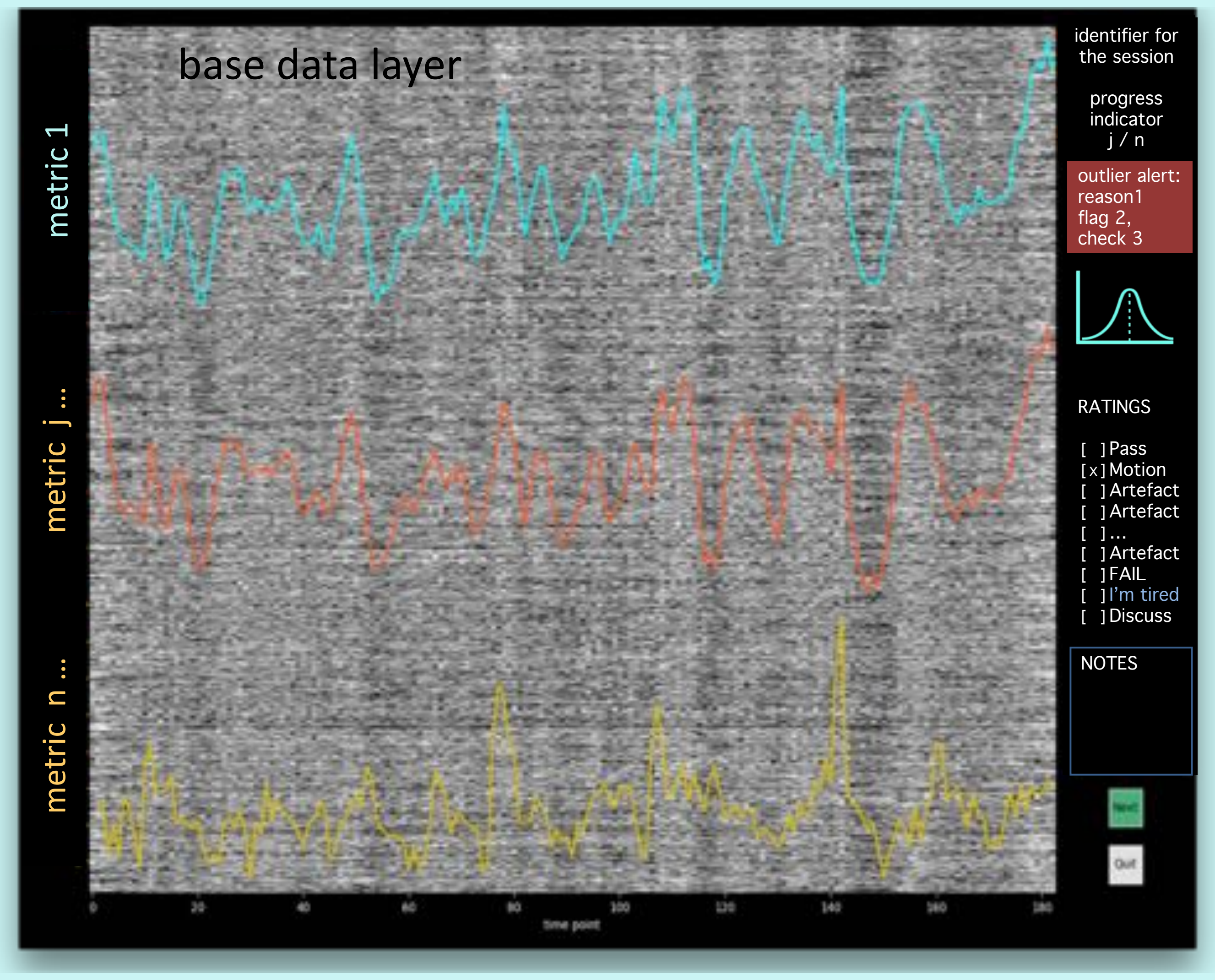
## Goal 1: Developing consensus

- Currently, there are many open questions and lack of consensus on how to perform proper QC
- For example, in defining levels of severity of artefacts – what is mild/moderate motion, and what is unacceptable?
- Consensus on what artefacts can be ignored in what pipelines?
- Developing a common QC protocol that identifies artefacts and rates their severity.

## Goal 2: Developing protocols

- A protocol could be requiring a specific series of checks to be performed depending on the use case
- To control the variability in review, each protocol could be restricted to a predefined set of visualizations, under chosen parameters, and requiring a comprehensive record of review activity
- A named and versioned protocol could be the basis, for reviewer and Editors, for communicating and evaluating the quality of input in a scientific study.

## Example Interface


base data layer
metric 1
metric j ...
metric n ...
time point
identifier for the session
progress indicator j / n
outlier alert: reason1 flag 2, check 3
RATINGS
[ ] Pass
[x] Motion
[ ] Artefact
[ ] Artefact
[ ] ...
[ ] Artefact
[ ] FAIL
[ ] I'm tired
[ ] Discuss
NOTES
Next
Quit