Reviewers' comments:

Reviewer #1: Summary - The motivation for the software article is to provide a framework, the Abstraction of Repository-Centric ANAlysis (Arcana), to extract common elements of repository centric workflow design that can be used for wide range of neuroimaging analyses. The authors have provided a clear and comprehensive explanation of the methods used in the work for the framework, with details on the workflow design, pipeline construction methodology, and its implementation to multi-modal studies. Using the Unified Modeling Language approach, a succinct explanation for the application of the Arcana framework in retrieving data from imaging repositories that contain data sets (plain directories, BIDS data sets and XNAT instances) has been shown. Analysis of Study instances has been clearly defined by the roles and functionalities of the repository, processor and environmental modules. The Arcana software framework has been made available to the users via GitHub for further testing and study.

Implementation of the framework has been demonstrated for neuroimaging analysis, for dMRI, T1Study, T2starStudy, as well as the multistudy. By separating analysis design from its application, Arcana framework allows workflows to be portable and applicable to datasets stored in various repositories.

Suggestions - The authors in the discussion section have indicated that modular pipeline and object oriented architecture promotes code reuse, and workflows can be used for neuroimaging datasets in multiple repositories. It might be interesting if the discussion section can be elaborated, to relate it to the broader discussion of the "FAIR" (findable, accessible, interoperable, and reusable) principles (https://www.ncbi.nlm.nih.gov/pubmed/26978244).

We thank the reviewer for the suggestion and have appended a paragraph to the discussion relating Arcana to the F.A.I.R guiding principles.

*Discussion, final paragraph:*

*The framework provided by Arcana should aid the development of workflows that are consistent with the F.A.I.R. guiding principles (Wilkinson et al., 2016). Each pipeline is accompanied by a text description and a list of citations to describe the techniques it employs. The extensibility and modularity of workflows in Arcana should encourage their reuse, and facilitate communal development of code repositories. Arcana analyses are interoperable between different repository and execution environments. In addition, the development of Banana is planned to closely follow the specification for BIDS derivatives (Gorgolewski, 2019) for interoperability with other BIDS apps.*

Specific consideration during revision -

1. In the methods section, 'data specification table' has been mentioned. It was not clear as to where the data specification table is stored, is it within repositories? Since data specification table is at the heart of each study, it would be helpful to have some clearer explanation about it.

The data specification table is an attribute of the Study Python class. We have replaced references to it as a “table” with “class attribute” to clarify this.

*Methods, Framework overview, 2nd paragraph:*

*The set of input data, the derivatives that can be derived from it, and methods that construct pipelines to derive the derivatives, are linked together by the data specification class attribute of the Study (Figure 1). …*

*Methods, Data and parameter specifications, 1st paragraph:*

*At the heart of each Study class is the data specification class attribute, which specifies the input and output data of the analysis, and all stages in between. …*

2. In the 'Data and parameter specification tables' section, in the third paragraph it was stated- "the distinction is fluid with derived specifications ----------------------- class is instantiated". Can the authors make this clearer as to exactly what is being stated with respect to data specification?

In order to clarify this section, we have reworded it to read

*Methods, Data and parameter specifications, 3rd paragraph:*

*… However, the distinction between input and derived data is somewhat blurred, as derivative specification entries can be passed input data when the class is instantiated. In addition, when subclassing a Study class or aggregating it in a MultiStudy class, derivative entries can be overridden by input entries, and vice-versa.*

3. In the workflow design section, the fifth paragraph states "interface traits", it was not clear, as to what these traits are being referenced to, is it related to pipeline construction?

We have reworked this paragraph and replaced references to “interface traits” with inputs (outputs) of the node’s input (output) specification.

*Methods, Pipeline constructor methods, 5th paragraph:*

*Node inputs and outputs are connected to each other, and to inputs and outputs of the pipeline, by providing inputs and outputs keyword arguments to the add method. Both arguments take a dictionary. The keys of the inputs dictionary correspond to trait names in the node’s input specification, whereas the keys of the outputs dictionary correspond to names of entries in the study data specification. The values of both dictionaries are 2-tuples. For pipeline inputs, values of the inputs dictionary consist of a name of an entry in the study data specification and the format the input data is expected in (i.e. a FileFormat for Fileset specifications or core Python type for Field specifications). For pipeline outputs, values of the outputs dictionary consist of a trait name in the node’s output specification and the format the output data is generated in. For input connections from other nodes, values of the inputs dictionary consist of a reference to the upstream node and the name of a trait in the upstream node’s output specification (output connections to other nodes are implied by input connections to the receiving nodes).*

4. In the implementing multi-modal studies section, first paragraph discusses "integration of data specification table" can the authors clarify whether integration is carried within repository modules?

This integration is defined within the MultiStudy Python class and is independent of the repository module. We hope that this point clearer after clarification that the data specification is stored in an attribute of the Python class and not in a repository (see response to Point 1).