Code Review Checklist

Most important before paper

Important every session

Optional

Usability	**Comments (at least Yes / No)**
Is the code easy to install/run? Are there setup instructions and a list of requirements?	Yes. A bit unclear from the readme whether on what input data it will work (ie., format data presumably starts from Charles formatting)
Is there an example script or a full pipeline that is easy to run and understand?	Not mentioned in the readme. There's no obvious .py file based on the naming.
Data preparation	
Are data loading and analysis implemented as separate steps? Ideal: have a data loader class	The data loading seems scripted into the loop of the main_parcels parcellation analysis. So no, although it is clear where the data is loaded and how (can be improved by pulling out these lines into a function - or do a class as suggested)
Is ALL data used available in the cluster	Not checked
Analysis & Plotting	
Are the different steps of the analysis clearly identified in the README?	More or less, but they can be improved. For example, (1) we don't know enough about the connectivity alg (is 'relabeling' a step that assigns the individual parcellations to a template's labels? (2) morphometry is a entirely orthogonal step? - how is this used? the visualizations seem to be for the connectivity results only from the readme?
Does the analysis code reflect what is described in the paper? If applicable	Not applicable
Is it clear what code is used to create each of the figures or panels in the paper?	Not applicable because there is not paper but there is readme info that gives a sense of what figures are

	produced
Code quality	
Project in periodically updated in github, gitignore, README	Yes. (there's only a few commits but that seems because the bulk of the coding was done over a few days)
Project structure: folders: data, notebooks, scripts, figures	It can be outlined in a tree diagram with more detail. For example, we tried to find the exact algorithm used for clustering, we had to find first what is in parcels.py, then go inside parcel_core and find out which functions perform In addition, the readme could describe an overview of the clustering methods so that we can imagine the following steps (e.g., what relabeling refers to) - and outline of post-parcellation steps could be useful too. In addition, it would help if the names were more informative. For example, parcel_core could clustering_code or something. Example code or a notebook for running through a full analysis may be useful
Is the code well organized (functions, classes, modules, settings, ... as applicable)?	Yes overall. It could use a bit more documentation, removing commented out code - some of the 'project structure' points above may apply here.
Are all functions and classes documented?	No, most of the functions are not thoroughly documented (maybe at least inputs and outputs in docstrings of the frontend scripts)
Are some values hardcoded?	They seem to be declared in a configurations part at the top of the main_parcels script - this is clear (good convention for such values is

	to declare them in ALL CAPS variable names)
Can any of the code be replaced by existing packages/functions?	It seems like it uses existing packages/functions where possible (but a closer look would be needed).
Are there any obvious optimisations that will improve performance?	No.
Is there any redundant code that should be removed/refactored?	There is a lot of commented-out code that can just be deleted (e.g., in parcels, parcel_main
Consistent, readable coding style (bonus points if. PEP8 for Python)	Yes, the code is readable
Variables names are self explanatory (eg no a, b, c etc)	The variable names are generally informative (with some exceptions; for e.g., global variable EPSILON in the parcel_main/connectivity.py
Are there any passwords in the repo or exposed in the code?	None found.
Is any identifying information unwillingly exposed?	None found.