Kirk's Preprocessing EEG Guide for PreciseKIDS and/or other EEG Data

Overview

Here's my guide for preprocessing EEG data. This guide will tell you how to go from raw data, straight off the collection computer, to EEG connectomes based on different FC methods and different frequency bands. If you follow the entire guide, the data will be warped to source space, and the connectome you generate will be based off the FC between brain regions, rather than between electrodes. This guide will perhaps still be useful to you even if you're doing a completely different EEG analysis.

For PreciseKIDS, we collected 64 channel EEG data, along with structural MRIs. I used the structural MRIs to transform the EEG data into source space, using a standard template of where electrodes should sit on a participant's head. If you want to do things "properly", you should also collect data of where the electrodes actually sit on a participant's head, rather than estimating it from a template.

For PreciseKIDS, data was collected across 4 visits, where 3 EEG scans were collected for each visit, along with a bunch of MRI/fMRI data. For each family, we collected data from both a parent and a child. For each participant, I picked a best MRI T1 (rather than using the MRI collected on the visit in question).

You could save yourself a lot of grief by instead analyzing data in sensor space. You could also save yourself a lot of grief by transforming the data to source space using a standard MRI template, rather than each participant's individual MRI.

The code I wrote is fairly preciseKIDS specific. You’ll have to change a lot of stuff if you, for e.g., have 256 EEG electrodes. But this guide should at least tell you what’s going on.

Preprocessing Overview

I've divided preprocessing into 3 parts: EEG preprocessing, MRI preprocessing, and final steps. EEG preprocessing and MRI preprocessing can be done at the same time, but both sets of steps need to be done before you can move onto the final steps.

Software: You’ll need a way of running a Python script. You’ll need to install various basic Python packages and MNE. For the MRI preprocessing, you’ll need Nipype and ANTs.

Part I: EEG Preprocessing

Note: I’ve baked into my scripts that I’m using 2 s non-overlapping epochs. If you want to do things differently, you’ll have to modify this line of code (and similar lines of code) in a lot of places:

mne.make\_fixed\_length\_epochs(raw\_downsampled,duration=2.0,preload=True)

Step 1) Get your EEG data

Ideally this data is already in BIDS naming convention, but is still in the .raw format. If your data is not .raw, you'll probably have to modify my scripts to convert them to load a non-.raw file. Or you can convert everything to .raw. Up to you.

If your data is .raw but not in the BIDS naming convention (as what would come out of the collection computer) you can use my newmakebids.py script. All the script does is rename files and sorts them to BIDS standard. The files themselves are unchanged. Note that this script is very much based on preciseKIDS data, so it'll almost certainly not work on other data. But maybe it's a good starting point.

If you’re not using MNE / my code, you might need a different file format.

Step 2) Run 01 bad channel investigator.py

This will frequency filter your data, downsample your data, and create a bunch of plots you can use to determine which channels are bad (if any). Bad channels must be determined manually

You will need to specify your frequency filter, downsample frequency, input directory, output directory, and probably your subject file name, which families to run, which participant ages to run, which sessions to run, which tasks to run. All these are specified at the top in the first few lines of code.

Note that the filtered/downsampled version of the data isn’t saved (yet), and 03 ICA.py and 05 finalize.py will redo these “initial” steps that are quick, minimizing the amount of files generated.

Step 3) Determine which (if any) of your channels are bad. This must be done manually, but the outputs of the previous script will help. You’ll notice a folder called “channelplots” that has some time course plots for each channel. The “repaired” plots are just with the worst noise censored out, so as to not distort the y axis. There’s also a document called badchanneldata.txt, that tells you things about each channel, such as its average correlation with other channels and average amplitude. If the average amplitude is too high or is very very low, or if the average correlation is very low, then it’s probably a bad channel (but check things visually). In that document:

Channels with most seconds dropped for all artifacts = how much of that channel’s time course has an unusually high amplitude

Channels with most seconds dropped for long artifacts = how much of that channel’s time course has an unusually high amplitude, but only for artifacts of a minimum length (so as to not include blinks, which are short duration and easily removed)

Specify the bad channels in a csv file that has two columns that looks like this:

|  |  |
| --- | --- |
| File | BadCh |
| sub-1973003C\_ses-2\_task-DORA3\_eeg.raw | none |
| sub-1973003C\_ses-1\_task-RX2\_eeg.raw | E23,E63 |
| sub-1973003C\_ses-3\_task-RX7\_eeg.raw | E62 |

The files can be listed in any order. Note, you must specify “none” if there are no bad channels

Step 4) Run 03 ICA.py

This will run an ICA on the data, allowing you to specify which independent components are bad due to eye or other artifacts. This must be run after you specify bad channels, because bad channels are removed from the data first, so they aren’t used to create ICs. What this program does is spit out a bunch of (hopefully) helpful information to classify bad ICs. Note that the suggested bad ICs are probably too aggressive, and I removed far fewer components than the default.

Note: If your ICA output looks like crap, that might mean you should specify more channels as bad channels

I’ve also seen some suggestions to remove really bad epochs before running ICA, so they don’t distort things too much. Things to try / keep in mind.

Step 5) Determine which (if any) of your independent components are bad. This must be done manually, but the outputs of the previous script will help.

How I determined bad ICs is I opened ICA\_0to15.pdf (and the similarly named files). Visually look for bad channels. “ICLabel” predicts if it’s eye/brain/other activity; remove eye components. The ICLabel score is the percent probability it’s a given component. I usually remove anything >60% chance of being eye, and anything >60% of being a bad channel, or anything that removes a lot of noise. It’s kinda subjective. I err on the side of keeping components.

I attempted to create some default bad ICs, and default eye ICs. It’s mostly based on the ICLabel scores. It sort of works, but it’s too aggressive at removing components. But it should at least give you a starting point of what to remove / not remove.

You can see how many noise epochs there are if you remove it. For example, if it says

“w/ oth eye: 841/0ep/3ch 🡪 441/0ep/0ch” that means

It’s a default eye component

If you remove the bad eye ICs, but not this one, there are 841 epochs marked as bad across all channels (this could be 100 bad epochs from 8 channels, and 41 bad epochs from a 9th channel. You can look at the plots to kinda see), 0 epochs are sufficiently bad across all channels, but 3 channels are bad. However, if you remove the bad eye ICs including this one, there are only 441 bad epochs, with no bad channels, and no bad epochs across all channels. So you should probably remove this one, cause there’s a huge improvement just by removing this one IC.

The ICs are roughly in order of how “important” they are, so there’s no point removing ICs numbered 30 or higher or so.

Specify the bad ICs in a csv file that has two columns that looks like this:

|  |  |
| --- | --- |
| File | BadIC |
| sub-1973003C\_ses-2\_task-DORA3\_eeg.raw | 0, 1, 4 |
| sub-1973003C\_ses-1\_task-RX2\_eeg.raw | 0,1,5,7,24,28 |
| sub-1973003C\_ses-3\_task-RX7\_eeg.raw | 0,4,13,38 |

Step 6) Run 05 finalize.py

This will save a new file with your specified frequency filter, downsampling, bad channels removed, bad ICs removed. If you want to know how to preprocess EEG data in general, not preciseKIDS data, try to read through this specific script, since the previous scripts didn’t produce anything other than plots to help decide bad channels/ICs (although this script also produces a bunch of ICA plots that are purely for QC and are not necessary)

Step 7) Confirm that your file is the correct length. For precisekids, you can do this by running expore\_file\_length.py. Further, check the file notes that the research assistant / whoever used when collecting data. You may need to remove some epochs at the start or the end. Mark any epochs you want to remove at the start/end in a csv file that looks like this:

|  |  |  |  |
| --- | --- | --- | --- |
| File | Omit | Drop\_start | Drop\_end |
| sub-1973003C\_ses-2\_task-DORA3\_eeg.raw | no | 0 | 0 |
| sub-1973003C\_ses-2\_task-RX4\_eeg.raw | no | 0 | 6 |
| sub-1973003C\_ses-3\_task-DORA6\_eeg.raw | no | 2 | 3 |

(Why would you do this? Maybe the person collecting the data had to manually start recording, or manually end recording, and you want to remove those questionable epochs. Hopefully for most of your recordings, you want to drop 0 at the start and 0 at the end.)

Part II: Structural Preprocessing

Some of these steps might be unnecessary. I had substantial difficulty getting skull/brain surfaces that worked, so I used a somewhat janky procedure, rather than using Freesurfer defaults. I’ll add commentary where necessary.

Step 1) For each participant, get 1 “best” T1w structural scan.

Step 2) Get an ANTs transformation matrix that warps the T1w into MNI space. You can do this by running Guinevere returns.py. You’ll need Nipype and ANTs installed, and you’ll need an MNI template. I used both a child and an adult MNI template.

Guinevere returns.py does the following:

-bias field correction

-brain extraction

-registration to MNI space

Make sure to specify the directory, the template directories, and you may need to specify the input (n4input), i.e., the name of the T1w file.

Note, this program takes like 3 hours per participant to run

Note, you won’t have to do this if you get better luck with the Freesurfer default surfaces (scalp surface, inner skull surface, outer skull surface, brain surface)

Step 3) Look for beads. Open your best T1w in FSLeyes or something. See if there’s a bead on the T1w. This might mess up the head surface Freesurfer calculates, so we’ll remove it. Find a voxel that contains the bead. Somewhere in the middle ish of the bead, doesn’t really matter. Run bead detector.py. This program works by assuming there’s a ‘gap’ of lower signal intensity in between the bead and the head. It keeps removing voxels around the bead voxel you chose, until the intensity gets too low, based on a threshold. For a janky program, it works fairly well. Try it with a low thres first, then if it fails, increase the thres. Start at 200, then 300, then 400, etc. Until it succeeds. If it never works… I don’t know what to tell you. Good luck! How do you know if it fails? It’ll tell you by saying “UH OH” or something. Just try again with a new threshold, it won’t save anything if that happens.

You might have to change the input\_base, output\_base, subjecthere, and beadlocation. Beadlocation is just the x,y,z coordinate of the bead voxel you chose.

Obviously you don’t have to do this step if there’s no beads on the T1w.

Step 4) Rename/sort the debeaded files into non-BIDS format:

e.g.: /Users/ivy/Desktop/Test\_EEG/Test\_MRI\_beadless\_nonBIDS/sub-1973003C\_ses-3/sub-1973003C\_ses-3\_T1w.nii.gz

Freesurfer seems to prefer them this way? I dunno, you could probably get Freesurfer to work in BIDS

Step 5) Run the debeaded files through FreeSurfer, preferably using ARC or some other high performance computer

Step 6) Run source localization script 1, to create (default) surface files. This takes about 20 minutes.

Step 7) Run source localization script 2, to set fiducial points. This has to be done manually.

Step 8) When you looked at the head surface in step 7, if you’re lucky, the surface of the head looks fantastic. If so, proceed to step 10. If you’re unlucky, you may have noticed that it looks wonky in some way, such as ‘horns’ or ‘antlers’ coming off of the surface. This can be due to “fog” (noise/artifact) around the head in the T1w scan. Arguably my jankiest program made is defogger.py. It lets you pick voxels that you think are the center of unusually high signal intensity surrounding the head, in what appears to be a cloud or something surrounding the person in the scanner. The program tries to remove this fog, so when you rerun Freesurfer, Freesurfer doesn’t get confused and think the participant has skin flaps. It doesn’t work very well, so you have to pick lots of voxels. You can try to fiddle with the settings, and run it multiple times. I make no guarantee it’ll help you.

Step 9) If necessary, rerun Freesurfer, now that you’ve removed the fog. Rerun steps 6-9 until things look non-stupid.

Step 10) Run source localization script 3, to align the MRI and EEG. This creates a transform file to align the two, and a stretch factor file (the default EEG electrode position montage is probably too big/small for the given person, especially if it’s a kid). You’ll have to specify the electrodefile, which specifies the standard positions of all the electrodes. Note that if you actually have real information of where the electrodes are (rather than estimates based on AdultAverageNet64), there’s a better way to align the MRI and EEG, but I have no information on that procedure. Read MNE tutorials.

Step 11) Run source localization script 4, to check the output of step 10. The virtual cap should sit nicely on the head. Note any electrodes that sit in a place where the surface is really bad (this is rare, and usually a cheek electrode when a participant has had major dental work). We’ll want to exclude those in a later script so just write down the bad electrodes for now.

What happens if things look bad? I don’t know. Try rerunning script 3? Maybe there’s some settings you can change? I never had that problem / I trusted the outputs.

Step 12) In the Freesurfer folder (where you have the folders bem, label, mri, etc) create a new folder “ANTS\_fun”. Into this folder copy outputs from step 2:

transformT1wFltmeantoMNIinverse.h5

transformT1wFltmeantoMNIinverse.nii.gz

This is used to create new skull surfaces, cause the Freesurfer default ones suck for some reason (for kids, anyway)

You won’t have to do this, if Freesurfer cooperates with you and creates non-stupid surfaces

Step 13) Run source localization script 5. This is the stupid script that makes custom surfaces. If you’re lucky, you can avoid doing this and just use the default scalp/skull/brain surfaces. How do you know if you’re lucky? Visually check the surfaces and see if they look okay and don’t overlap with each other. I would probably either use the defaults for all participants, or the Kirk altered surfaces for all participants. Keep it consistent.

If you’re wondering how the script works. Ugh. But at a very high level, it looks at the Freesurfer default brain mask, and MNI template brain masks, and creates a combined very tight brain mask. Then it checks the inner skull surface, and moves it to make sure it doesn’t overlap with the brain surface. Then it checks the outer skull surface, and moves it closer to the scalp surface. This should create enough room for everything to look reasonable and for things to run.

If you think this sounds stupid, the MNE suggestion in how to fix bad surface files is to open Blender and manually fix them. I prefer my automated approach.

If you’re unlucky and have to run this script… you’ll need to specify mnimask\_c and mnimask\_p, your child and adult masks of brains in MNI space. You’ll note I created a custom adult mask to match the child one (i.e., they’re roughly equally liberal in classifying brain and non brain). Cause why would this be easy for Kirk??

Step 14) Run source localization script 6. This checks the output of script 5. You’re most interested that the following layers make sense:

Outer\_skin\_orig.surf

Outer\_skull\_expanded.surf

Inner\_skull\_final.surf

Brain\_combined.surf

The brain should be inside the inner skull, the inner skull should be inside the outer skull, the outer skull should be inside the skin. And the layers should align reasonably well with the T1.

What happens if things look bad? There’s a billion lines of code in script 5 you can fiddle with. Maybe something will make it look better. Hopefully you can just run script 5 again and it’ll somehow work.

Part III: Finals Steps

Hooray, you got this far. This took Kirk several painful months, so hopefully you’re making better time than him.

Step 1) Remember how back in Step 11 of Part II we wrote down bad electrodes? Open the script Source localization 7 and specify these under specialchange. If there aren’t any bonus bad electrodes, then you don’t have to do anything.

Step 2) Run source localization script 7. This creates the forward solution for warping the EEG data into source space. It takes ~10 minutes per file

Step 3) Run source localization script 8. You’ll have to specify the location of the manual adjustment csv (Step 7 of Part 1). You’ll also have to specify which parcellation to use, which frequency bands to use, and which FC measures to use.

If you want to use a different parcellation, or different settings for getting things into source space, this is the program to edit.

Note that no matter what FC measures you specify, it’ll also spit out phase slope index connectomes. You could edit the code to stop that, but I’m kinda lazy myself.

Step 4) Pat yourself on the back, you’ve successfully generated some EEG connectomes! What do you do with them? That’s for you to decide.